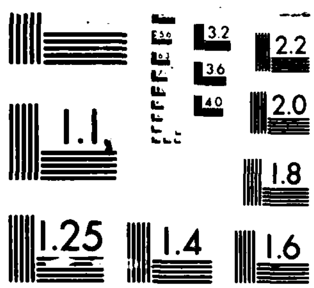
AD-A178 542 A FAMILY OF ALGORITHMS FOR THE ESTIMATION OF THE PARAMETERS OF THE STABLE..(U) RENSSELAUER POLYTECHNIC INST TROY NY SCHOOL OF MANAGEMENT T A DELEHANTY ET AL. 1/1

UNCLASSIFIED 1986 WP-37-86-P34 ARO-18072.25-MA F/G 12/1 NL

END
DATE
FILMED
5-87


3.2
3.6
4.0
2.2
2.0
1.1
1.8
1.25
1.4
1.6

Rensselaer Polytechnic Institute
School of Management

Working Paper No. 37-86-P34

A FAMILY OF ALGORITHMS FOR THE ESTIMATION OF THE PARAMETERS OF THE STABLE LAWS AND THE PARAMETERS OF ATTRACTING STABLE LAWS

by

T. A. Delehanty
and
A. S. Paulson

School of Management
Rensselaer Polytechnic Institute
Troy, NY 12180-3590
(518)266-6586

DESCRIPTION AND PURPOSE

Potential application of the stable laws has long been hindered by the unavailability of generally available, well-documented algorithms. This paper removes this deficiency by presenting an algorithm for estimation of stable law parameters, with the goal of facilitating the application of stable laws in modeling and inference frameworks. The stable laws have steadily increased in importance to the statistical community since the paper of Mandelbrot (1963). Their role as the only laws possessing domains of attraction makes the stable laws an appealing probabilistic model, and they are capable of modeling a wide range of skewness, heavy tailedness, and central peakedness. Procedures for estimation of stable law parameters have been described by Mandelbrot (1963), DuMouchel (1971), Fama and Roll (1971), Paulson, Holcomb, and Leitch (1975), Koutrouvelis (1980,1981), Feuerverger and McDunnough (1981a,1981b), and Brockwell and Brown (1981). Because of the intractability of stable densities, attention has centered in recent years on Fourier-based procedures, using the empirical characteristic function. Such procedures should have an adaptive nature (Paulson, Holcomb, and Leitch, 1975; Paulson, Delehanty, and Brothers, 1982; Paulson and Delehanty, 1982).

~~We~~ This document presents an iterative and adaptive algorithm for joint estimation of stable law parameters, using the empirical characteristic function. The algorithm is flexible in that either of two procedures may be selected, and subsets of the parameters may be allowed to vary freely, with others constrained or held constant. The statistical rationale for the procedures is described in the companion paper by Paulson and Delehanty (1982). The algorithm may also be used to provide informal estimates of the parameters

A FAMILY OF ALGORITHMS FOR THE ESTIMATION OF THE PARAMETERS
OF THE STABLE LAWS AND THE PARAMETERS
OF ATTRACTING STABLE LAWS

by

T. A. Delehanty

and

A. S. Paulson*
Rensselaer Polytechnic Institute

Key words: stable laws, parameter estimation, adaptive estimation, empirical characteristic function, domains of attraction, sensitivity analysis, nonlinear optimization, constrained estimation

LANGUAGE: FORTRAN 66

*Research supported in part by U.S. Army Research Office under contract DAA G29-81-K-0110.

of the stable law to which a sample distribution is attracted.

THEORY AND NOTATION

Nondegenerate stable random variables X may be defined by the characteristic function

$$\phi(u) = E(\exp iuX) = \exp\{iu\mu - |\sigma u|^{\alpha}(1+i\beta \operatorname{sgn}(u)\ \chi(u,\alpha))\}, \tag{1}$$

where $i^2 = -1$, $0 < \alpha \le 2$, $|\beta| \le 1$, $\sigma > 0$, and

$$\chi(u,\alpha) = \begin{cases} \tan\frac{\pi\alpha}{2}, & \alpha \ne 1 \\ \frac{2}{\pi}\log|u|, & \alpha = 1. \end{cases} \tag{2}$$

Here α, the characteristic exponent, is a measure of heavy tailedness and central peakedness, β is a skewness measure, σ is a scale parameter, and μ is a location parameter unless ($\alpha=1$, $\beta\ne 0$), when the function of location parameter is assumed by $\mu + \frac{2}{\pi}\beta\sigma\log\sigma$. The only stable laws whose densities are expressible in closed form are the Gaussian ($\alpha=2$, $\beta=0$), the Cauchy ($\alpha=1,\beta=0$), and the reciprocal of a χ^2 variate on one degree of freedom ($\alpha=\frac{1}{2},\beta=-1$).

Let $X_1,\ldots,X_n$ be a stable random sample. The empirical characteristic function is

$$\phi_n(u) = n^{-1}\sum_{j=1}^{n} \exp(iuX_j). \tag{3}$$

Let $\psi(u) = \operatorname{Re}\phi(u) + \operatorname{Im}\phi(u)$, $\psi_n(u) = \operatorname{Re}\phi_n(u) + \operatorname{Im}\phi_n(u)$. Estimators interior to the parameter space can be viewed as zeros of the systems

<u>Formulation A</u>

$$\sum_{j=1}^{q} \frac{\partial\psi(u_j)}{\partial\theta}(\psi(u_j) - \psi_n(u_j))w_j = 0, \tag{4}$$

or

Formulation B

$$\sum_{j=1}^{q} \sum_{k=1}^{q} \frac{\partial \psi(u_j)}{\partial \theta} K^{jk} (\psi(u_k) - \psi_n(u_k)) w_j w_k = 0, \tag{5}$$

for $\theta = \alpha, \beta, \sigma, \mu$. The grid $\{u_j | j=1,\dots,q\}$ is symmetric about zero but does not include the origin, and K^{jk} denotes the j,k element of the inverse matrix $(K_{jk})^{-1}$, where

$$\begin{aligned} K_{jk} &= n \operatorname{cov}(\psi_n(u_j), \psi_n(u_k)) \\ &= \operatorname{Re} \phi(u_j-u_k) + \operatorname{Im} \phi(u_j+u_k) - \psi(u_j)\, \psi(u_k). \end{aligned} \tag{6}$$

The weights $\{w_j | j=1,\dots,q\}$ also depend on the parameters $\alpha, \beta, \sigma, \mu$, and are described in the Numerical Method section. Both Formulations A and B represent modified, weighted χ^2 minimum procedures, corresponding to the respective objective functions

$$\text{A:} \qquad S_n = \sum_{j=1}^{q} (\psi(u_j) - \psi_n(u_j))^2 w_j, \tag{7}$$

$$\text{B:} \qquad Q_n = \sum_{j=1}^{q} \sum_{k=1}^{q} (\psi(u_j) - \psi_n(u_j))\, w_j\, K^{jk}\, w_k(\psi(u_k)-\psi_n(u_k)). \tag{8}$$

The following points are critical for practical application:

1) The shapes of ψ and ψ_n are highly dependent on location and scale parameters, and so should be standardized;

2) The estimators are improved by making the gridpoints and weights depend on α and β;

3) Since the procedures are adaptive ($\{u_j\}$, $\{w_j\}$ and $\{K_{jk}\}$ depend on unknown parameters), algorithms must be iterative;

4) Since α, β and σ are always constrained, each iteration involves solution of a nonlinear optimization problem with variable bound constraints.

Our procedures may therefore be summarized as follows, where a tilde indicates estimators, their values, or adaptively standardized quantities.

Begin with initial guesses for the parameters. At each iteration, compute and save $\{u_j\}$, $\{w_j\}$, possibly $\{K^{jk}\}$, and standardized empirical characteristic function values $\{\tilde{\psi}_n(u_j)\}$, based on the latest $(\tilde{\alpha},\tilde{\beta})$. The objective S_n or Q_n is then minimized (an "optimization subproblem"), and cumulative location and scale estimates $(\tilde{\ell},\tilde{s})$ are updated. Iteration stops when values of σ and μ minimizing S_n or Q_n are acceptably close to unity and zero, respectively.

Estimators whose values are not on a bound are asymptotically multivariate Gaussian distributed. The asymptotic covariance matrices $\underset{\sim}{\Sigma}$ of the estimators are derived in Paulson and Delehanty (1982). The basic formula is

$$\underset{\sim}{\Sigma}_i = \underset{\sim}{H}_i^{-1} \underset{\sim}{V}_i \underset{\sim}{H}_i^{-1}, \qquad i = A,B. \tag{9}$$

There are two particularly appealing ways to approximate $\underset{\sim}{\Sigma}$. In "approximation (i)", expectations are approximated from the data: $\underset{\sim}{H}$ is computed by differencing the objective at the final optimum, and

$$\underset{\sim}{V}_A = 4\underset{\sim}{D}^T\underset{\sim}{\tilde{K}}\underset{\sim}{D}, \tag{10}$$

$$\underset{\sim}{V}_B = 4\underset{\sim}{D}^T\underset{\sim}{K}^{-1}\underset{\sim}{\tilde{K}}^{(w)}\underset{\sim}{K}^{-1}\underset{\sim}{D}. \tag{11}$$

Here

$$D_{j\theta} = \frac{\partial\tilde{\psi}(u_j)}{\partial\theta} w_j, \quad (12)$$

θ ranging over the parameters free of bounds,

$$\tilde{K}_{jk} = n^{-1} \sum_{m=1}^{n} (\tilde{\psi}(u_j) - \cos u_j\tilde{x}_m - \sin u_j\tilde{x}_m)(\tilde{\psi}(u_k) - \cos u_k\tilde{x}_m - \sin u_k\tilde{x}_m), \quad (13)$$

and

$$\tilde{K}^{(w)}_{jk} = \tilde{K}_{jk} w_j w_k. \quad (14)$$

By location and scale invariance, $(\tilde{\sigma},\tilde{\mu})$ are set to (1,0) during these computations, and $\underset{\sim}{\Sigma}$ scaled. In "approximation (ii)", expectations are calculated analytically, so $\underset{\sim}{K}$ replaces $\tilde{\underset{\sim}{K}}$ in (10), (11), and (14), and factors of 2 are omitted. The expected Hessian has elements

$$H_{A\theta\theta'} = \sum_{j=1}^{q} \frac{\partial\tilde{\psi}(u_j)}{\partial\theta} \frac{\partial\tilde{\psi}(u_j)}{\partial\theta'} w_j, \quad (15)$$

$$H_{B\theta\theta'} = \underset{\sim}{D}^T \underset{\sim}{K}^{-1} \underset{\sim}{D},$$

where θ and θ' range over free parameters.

To analyze domains of attraction, we use what we refer to as the k-sum procedure. If k is a positive integer, the power

$$\phi_n^k(u) = n^{-k} \sum_{j_1=1}^{n} \cdots \sum_{j_k=1}^{n} \exp(iu(x_{j_1} + \cdots + x_{j_k})) \quad (16)$$

is the characteristic function corresponding to the k^{th} convolution power of the empirical distribution function $F_n(x)$, and can be interpreted as empirical characteristic function of all possible k-sums $\{x_{j_1} + \cdots + x_{j_k}\}$, sampling with replacement from $F_n(x)$. We add real and imaginary parts and standardize, giving $\tilde{\psi}_n^k(u)$, say, and estimate $(\tilde{\alpha}_k, \tilde{\beta}_k, \tilde{\sigma}_k, \tilde{\mu}_k)$ for different values of k. If the sample distribution is

attracted to a stable law with parameters $(\alpha,\beta,\sigma,\mu)$, the sequence of normalized estimators $(\tilde{\alpha}_k, \tilde{\beta}_k, \tilde{\sigma}_k/k^{1/\tilde{\alpha}_k}, \tilde{\mu}_k/k)$, for reasonable values of k, should approach $(\alpha,\beta,\sigma,\mu)$. In particular, a rapid rise in $\{\tilde{\alpha}_k\}$ may indicate that a stable model is not appropriate, a possible alternative being a mixture of finite variance components with differing scale parameters.

The k-sum procedure can thus be used in a sensitivity analysis, to examine how well the data support the stability assumption. Other possible tools for sensitivity analysis are varying the mechanism (to be described below) underlying the weights $\{w_j\}$, and comparing approximations (i) and (ii) of the estimated asymptotic covariance matrix, provided n is large enough for approximation (i) to be accurate.

NUMERICAL METHOD

The main computational task required is solution of bound-constrained nonlinear optimization problems. Although Formulations A and B lead to nonlinear least squares problems, current algorithms for nonlinear least squares do not allow constraints (Hiebert, 1981). Numerical Algorithms Group (NAG) subroutine E04KBF (NAG, 1981) is used for optimization. E04KBF is a quasi-Newton procedure, requiring an objective function and analytical first partial derivatives. It is substantially faster than the gradient projection routine used by Paulson, Holcomb and Leitch (1975), although the latter is very reliable. The other complicated numerical procedure required is inversion of a positive definite symmetric matrix ($\underset{\sim}{K}$, $\underset{\sim}{H}$ or $\underset{\sim}{\tilde{H}}$), for which NAG subroutine F01ABF is used. Various NAG utility procedures are also used, see Auxiliary Algorithms. The use of NAG

procedures inhibits transportability in that the algorithm, as presented, is only usable at installations having the NAG Library. However, listings of rapid, high-quality algorithms for constrained optimization have not appeared in the literature (see Chambers, 1977, pp. 159-160; the situation described there has not improved). Given that E04KBF is used, reliance on additional NAG Library procedures is expedient.

We require a minimum of q=20 gridpoints $\{u_j\}$, and prefer q=20 or 40, since they are reasonable values in practice, and have been tested extensively. Only the positive gridpoints are explicitly required, due to symmetry of the grid and the Hermitian property of characteristic functions. They are computed as follows: An endpoint U is chosen as 3, $\tilde{\alpha} \geq 1.8$; 3.3, $1.8 > \tilde{\alpha} \geq 1.7$; 3.6, $1.7 > \tilde{\alpha} > 1$; 5, $\tilde{\alpha}=1$; 4, $1 > \tilde{\alpha} \geq .9$; 5, $.9 > \tilde{\alpha} \geq .8$; 7, $.8 > \tilde{\alpha} \geq .6$; 10, $\tilde{\alpha} < .6$. An inner number I of points is selected close to the origin: I=2 if q<30 and 3 if q≥30, 1 being subtracted if $\tilde{\alpha} \leq .5$. The inner I points are spaced as follows: if $\tilde{\alpha}>1$, ½ the "α-optimal" values of Feuerverger and McDunnough (1981b) for the nearest (larger) α are used; if $\tilde{\alpha} \leq 1$, the first I points giving q/2 equal increments of $\log (u + \alpha^{*3})$ between 0 and U are multiplied by $\frac{1}{4} \alpha^{*2}$ $(\alpha^{*} = \max(\tilde{\alpha}, .3))$. The remaining points are logarithmically spaced out to U: if $\tilde{\alpha}>1$, the function $\log (1 + u/2)$ is used, and if $\tilde{\alpha} \leq 1$, $\log(u + \alpha^{*3})$ is used. This rather complicated ad hoc scheme was developed through graphical inspection of $\tilde{\psi}(u)$ and $\tilde{\psi}_n(u)$, comparisons of asymptotic efficiencies, and parameter estimation for real and simulated data. No claims of optimality are made, but the scheme provides high efficiencies if efficiency is preferred, or good matches between $\tilde{\psi}$ and $\tilde{\psi}_n$ if curve fitting is preferred. The point of stratified and logarithmic spacing is

to emphasize u values near the origin. Details when $\tilde{\alpha} \le 1$ reflect the fact that $\psi(u)$ has a sharp cusp near the origin, but decays slowly. The stepwise nature of the scheme is not deemed a serious drawback.

The weights $\{w_j\}$ are computed as follows:

Under Formulation A,

$$w_j = \frac{|\phi(u_j)|^{2\lambda}}{K_\tau(u_j,u_j)} = \frac{\exp(-2\lambda|u_j|^\alpha)}{K_\tau(u_j,u_j)} \quad , \tag{17}$$

and under Formulation B,

$$w_j = |\phi(u_j)|^\lambda = \exp(-\lambda|u_j|^\alpha), \tag{18}$$

where λ and τ are supplied by the user, $0 \le \tau \le 1$,

$$K_\tau(u,u) = 1 + \tau(\mathrm{Im}\ \tilde{\phi}(2u) - \tilde{\psi}^2(u)), \tag{19}$$

and λ is recommended nonnegative. Rationale for these weights, and some corresponding asymptotic efficiencies, are in Paulson and Delehanty (1982). We recommend $\tau=1$ under Formulation A. Under Formulation B, it is convenient to let $\tau \ge 0$ represent a fraction of the average diagonal element by which to inflate K, giving a matrix we shall call A. We have only found this inflation necessary if $\tilde{\alpha}$ is very close to two, when $\tau = 0.01$ suffices.

To use the quantity λ as a tool for sensitivity analysis, we interpret it as a damping factor, lessening the effects of noise in $\tilde{\psi}_n(u)$ for larger $|u|$. If the data are truly stable and the sample size is fairly large (say 150 or more), estimates should change little as λ varies, say, from 0 to 1. Large discrepancies in the estimates for different values of λ indicate problems with the data or the stable assumption or both. It may not be easy to isolate the difficulty but further study is

definitely required.

For the k-sum procedure, $k>1$, the situation regarding gridpoints and weights changes. Tests so far indicate that when $\tilde{\alpha}>1$, Formulation A, with gridpoints equispaced from 0 to U, gives better results than "efficient configurations" used for $k=1$. Reasons for this are unclear. A possible explanation is that when $\tilde{\alpha}>1$ and $k>1$, $\tilde{\psi}_n^k(u)$ is so smooth that estimation is practically equivalent to deterministic curve fitting, and implicit or explicit emphasis on gridpoints near the origin neglects important curvature for large $|u|$. Accordingly, when $k>1$ and $\tilde{\alpha}>1$, we equispace gridpoints and set all weights to 1. When $\tilde{\alpha}\leq 1$, $\tilde{\psi}_n^k(u)$ has a sharp cusp near the origin and remains a jagged curve as k increases, due to the presence of very large observations. In this situation, we set all weights to 1 and use basically the same gridpoint scheme as when $k=1$, omitting only multiplication of the inner points by $\alpha^{*2}/4$. In either case, Formulation A is recommended.

An important question is how large k may be taken. Equation (16) suggests that we cannot expect to take k arbitrarily large. There seems to be a tendency for $\tilde{\alpha}$ to increase and $\tilde{\beta}$ to drift if k is too large, though this may be partially due to suboptimal gridpoints or weighting. It appears that when n is large, say 500 or more, k may safely be taken up to 20. Care is required for smaller n, and when α is small or very near two.

Implicit standardization is carried out as follows. Let k be a positive integer, and $(\ell,\tilde{s})$ cumulative location and scale estimates. Then

$$\tilde{\psi}_n^k(u_j) = \rho_{jk}(\cos \gamma_{jk} + \sin \gamma_{jk}), \tag{20}$$

where

$$\rho_{jk} = |\phi_n(u_j/\tilde{s})|^k \tag{21}$$

and

$$\gamma_{jk} = k \arg\phi_n(u_j/\tilde{s}) - \tilde{\ell}u_j/\tilde{s}. \tag{22}$$

No problems of principal values arise, and complex arithmetic is not used. The FORTRAN mathematical library function ATAN2 computes arguments.

The estimator $\tilde{\alpha}$ may be bounded in (closed) subintervals of $[\delta,1-\varepsilon]$, $[1,1]$, or $[1+\varepsilon,2]$ unless $\tilde{\beta}$ is fixed at 0, when $[\delta,2]$ is possible (δ and ε are small positive numbers), while $\tilde{\beta}$ may be bounded in subintervals of $[-1,1]$. Estimators $\tilde{\sigma}$ and $\tilde{\mu}$ may be constrained arbitrarily in$[\delta,\infty)$ and $(-\infty,\infty)$, respectively, unless $\tilde{\alpha}$ is fixed at 1 and $\tilde{\beta}$ is not fixed at 0, when $\tilde{\mu}$ cannot be constrained. Bounds on σ and μ are internally set for use in subproblems. These bounds must be wide enough to allow the "true values" to be found, but narrow enough to deter straying into undesirable regions, particularly $\sigma\to\infty$, $|\mu|\to\infty$. The <u>ad hoc</u> bounds of $[-5,5]$ for μ and $[0.2,5]$ for σ work well in practice. If $\tilde{\sigma}$ or $\tilde{\mu}$ are initially constrained, their internal bounds are adaptively modified, see the Algorithm for description.

Initial guesses for the parameters are required. We do not find their specification particularly important, provided $\tilde{\alpha}$ is on the correct side of 1 in the nonsymmetric case. We have used the median and semi-interquartile range as guesses for $\tilde{\mu}$ and $\tilde{\sigma}$, and averages of upper and lower bounds for $\tilde{\alpha}$ and $\tilde{\beta}$. If $\tilde{\alpha}$ is anticipated less than 1.2, say, it is

worthwhile to put more effort into initial guesses, since fewer iterations will be required (the semi-interquartile range will overestimate σ, and if α is near but different from 1, the median is nearer $\mu - \frac{2}{\pi}\beta\ \sigma\ \log\sigma$ than μ).

Convergence is judged by a tolerance on subproblem solutions, $\max(|\sigma^{(m)}-1|, |\mu^{(m)}|)$, (or $\max(|\alpha^{(m)} - \alpha^{(m-1)}|, |\beta^{(m)} - \beta^{(m-1)}|)$ if $\tilde{\sigma}$ and $\tilde{\mu}$ are fixed), with a maximum allowable number of iterations. Attainable tolerances depend on n, but more strongly on the underlying parameters. If $\tilde{\alpha}$ is near two, $\tilde{\psi}_n$ is very smooth and stringent tolerances such as 10^{-6} may be attained. If $\tilde{\alpha}\le 1.2$, $\tilde{\psi}_n$ has many small oscillations due to large observations, and, especially for smaller samples, it may be preferable to terminate after a fixed number of iterations. Good estimates are generally obtained within five iterations, fewer if initial guesses are good; if stringent tolerances are required, or for difficult problems (skewed distributions with $0.9\le\alpha\le 1.1$) more may be required. Convergence is typically slower under Formulation B, since the weighting mechanism is more complicated.

Approximation of asymptotic covariance matrices requires little description. We note that for approximation (i) and the q values we use, it is faster to define a vector

$$\delta_j^T = (\tilde{\psi}(u_1) - \cos u_1\tilde{x}_j - \sin u_1\tilde{x}_j, \cdots, \tilde{\psi}(u_q) - \cos u_q\tilde{x}_j - \sin u_q\tilde{x}_j), \tag{23}$$

and cumulate

$$\tilde{\underset{\sim}{V}} = 4n^{-1} \sum_{j=1}^{n} (\underset{\sim}{D}^T\delta_j)(\underset{\sim}{D}^T\delta_j)^T \tag{24}$$

under Formulation A, or

$$\tilde{V} = 4n^{-1} \sum_{j=1}^{n} (D^T A^{-1} \delta_j)(D^T A^{-1} \delta_j)^T \tag{25}$$

under Formulation B, than to cumulate $\tilde{K}$. The matrix D is computed by the function/gradient subroutine. E04KBF returns an approximate Hessian, which could conceivably be used for $\tilde{H}$ in approximation (i). Rather often, however, E04KBF will terminate with its failure indicator set to 3 and the Hessian set to the identity matrix, even though the optimum may be reliable. It is therefore simpler to compute $\tilde{H}$ by differencing. The following procedures is used: Set an initial Hessian to 0, and the steplength to 10^{-3}. Successively divide the steplength by $\sqrt{10}$ and approximate the Hessian by differencing; three-point differencing for the diagonal, and four-point for off-diagonal elements. Compare elements of successive approximations by maximum relative or absolute differences, according as the element of the latest approximation exceeds 1 in absolute value or not. A tolerance of 10^{-6} is used for this convergence criterion. If convergence has not occurred with a steplength of 10^{-5}, the result with steplength 10^{-4} is used.

Approximation (i) of the asymptotic covariance matrix is rather expensive to compute. It should not be computed for smaller sample sizes, as it implicitly involves estimation of $\frac{1}{2}q(\frac{1}{2}q+1)$ covariances.

Following is an informal description, in Algorithm form, of the basic routine STABLE. Approximate asymptotic covariance matrices may also be computed, but this presents no logical difficulties, so is omitted.

Algorithm

Produces estimates $(\tilde{\alpha},\tilde{\beta},\tilde{\sigma},\tilde{\mu})$ for k-sums $(k\geq 1)$, based on a sample $(x_1,\ldots,x_n)$.

Input parameters: k, n, $\{x_j\}$, q, λ, τ, Formulation (A or B), convergence tolerance ε, maximum number M of iterations, and flags whether $\tilde{\sigma}$ and $\tilde{\mu}$ are constrained.

Input/output parameters: $(\tilde{\alpha},\tilde{\beta},\tilde{\sigma},\tilde{\mu})$ are initial guesses on entry and estimates on exit, $(\alpha_L,\beta_L,\sigma_L,\mu_L)$ and $(\alpha_u,\beta_u,\sigma_u,\mu_u)$ are lower bounds. The (σ,μ) bounds are changed, but restored on exit. In the special case where $\tilde{\alpha}$ is fixed at 1, $\tilde{\mu}$, on entry, is the initial guess for location $\mu + \frac{2}{\pi}\beta\sigma\log\sigma$.

Auxiliary quantities $\tilde{\ell}$ and $\tilde{s}$ are cumulative location and scale estimates. Entry values of $(\sigma_L,\sigma_u,\mu_L,\mu_u)$ are stored in (b_1,b_2,b_3,b_4). On entry and exit, $(\tilde{\sigma},\tilde{\mu},\sigma_L,\mu_L,\sigma_u,\mu_u)$ are normalized.

<u>S1</u> [Initialize.]

Set $\tilde{\ell} \leftarrow k\tilde{\mu}$, $\tilde{s} \leftarrow k^{1/\tilde{\alpha}}\,\tilde{\sigma}$, $m \leftarrow 0$.

Save $(b_1,b_2,b_3,b_4) \leftarrow (\sigma_L,\sigma_u,\mu_L,\mu_u)$.

<u>if</u> $\tilde{\mu}$ is unconstrained <u>then</u> set $\mu_L \leftarrow -5$, $\mu_u \leftarrow 5$;

   <u>else</u> <u>if</u> $\tilde{\mu}$ is fixed <u>then</u> set $\mu_L \leftarrow \mu_u \leftarrow 0$;

      <u>else</u> set $\mu_L \leftarrow k\mu_L$, $\mu_u \leftarrow k\mu_u$.

<u>if</u> $\tilde{\sigma}$ is unconstrained <u>then</u> set $\sigma_L \leftarrow 0.2$, $\sigma_u \leftarrow 5$;

   <u>else</u> <u>if</u> $\tilde{\sigma}$ is fixed <u>then</u> set $\sigma_L \leftarrow \sigma_u \leftarrow 1$;

      <u>else</u> set $\sigma_L \leftarrow k^{1/\tilde{\alpha}}\,\sigma_L$, $\sigma_u \leftarrow k^{1/\tilde{\alpha}}\,\sigma_u$.

<u>S2</u> [Looping point for iteration; save adaptive quantities for subproblem.]

Increment $m \leftarrow m + 1$.

Save $\tilde{\alpha}^{(m-1)} \leftarrow \tilde{\alpha}$, $\tilde{\beta}^{(m-1)} \leftarrow \tilde{\beta}$.

<u>if</u> $\tilde{\sigma}$ is constrained but not fixed <u>then</u> set $\sigma_L \leftarrow \max(0.2, b_1/\tilde{s})$,

$\sigma_u \leftarrow \min(5, b_2/\tilde{s})$.

<u>if</u> $\tilde{\mu}$ is constrained but not fixed <u>then</u> set $\mu_L \leftarrow \max(-5, (b_3-\tilde{\ell})/\tilde{s})$,

$\mu_u \leftarrow \min(5, (b_4-\tilde{\ell})/\tilde{s})$.

Set $\tilde{\sigma} \leftarrow 1$, $\tilde{\mu} \leftarrow 0$.

Compute and save positive gridpoints $\{u_j \mid j = q/2+1, \ldots, q\}$,
weights $\{w_j \mid j=1,\ldots,q\}$, and standardized empirical characteristic function values $\{\tilde{\psi}_n^k(u_j) \mid j=1,\ldots,q\}$.

<u>if</u> Formulation B <u>then</u> compute and invert A.

<u>S3</u> [Subproblem.]

Solve the optimization problem

$$\min \sum_{j=1}^{q} (\psi(u_j) - \tilde{\psi}_n^k(u_j))^2 \, w_j \qquad \text{(Formulation A)}$$

or

$$\min \sum_{i=1}^{q} \sum_{j=1}^{q} w_i(\psi(u_i) - \tilde{\psi}_n^k(u_i)) \, A^{ij}(\psi(u_j) - \tilde{\psi}_n^k(u_j))w_i \quad \text{(Formulation B)},$$

yielding new $(\tilde{\alpha},\tilde{\beta},\tilde{\sigma},\tilde{\mu})$.

<u>S4</u> [Update and test convergence.]

Set $\tilde{\ell} \leftarrow \tilde{\ell} + \tilde{s}\,\tilde{\sigma}$.

<u>if</u> $\tilde{\alpha}=1$ <u>then</u> set $\tilde{\ell} \leftarrow \tilde{\ell} + \frac{2}{\pi}\,\tilde{s}\,\tilde{\beta}\,\tilde{\sigma}\log\tilde{\sigma}$.

Set $\tilde{s} \leftarrow \tilde{s}\,\tilde{\sigma}$.

<u>if</u> $\tilde{\sigma}$ and $\tilde{\mu}$ are fixed <u>then</u> error $\leftarrow$ max $(|\tilde{\alpha}-\tilde{\alpha}^{(m-1)}|, |\tilde{\beta}-\tilde{\beta}^{(m-1)}|)$;

<u>else</u> error $\leftarrow$ max $(|\tilde{\sigma}-1|, |\tilde{\mu}|)$.

<u>if</u> error $\geq \varepsilon$ and $m<M$ <u>then</u> go to <u>S2</u>.

<u>S5</u> [Final estimates.]

Set $(\sigma_L, \sigma_u, \mu_L, \mu_u) \leftarrow (b_1, b_2, b_3, b_4)$.

<u>if</u> $\tilde{\alpha}=1$ <u>then</u> set $\ell \leftarrow \ell - \frac{2}{\pi}\tilde{\beta}\,\tilde{s}\log\tilde{s}$.

Set $\tilde{\mu} \leftarrow \ell/k$, $\tilde{\sigma} \leftarrow \tilde{s}/k^{1/\tilde{\alpha}}$.

STRUCTURE

SUBROUTINE STABLE (X,N,MODE,KSUM,XLAM,TAU,NPTS,TOL,MAXIT,XL,XB,XH,NPAR,
ISCLBD,LOCBND,ICOV,VCV1,VCV2,WORK,LWORK,IWORK,LIWORK,IFAULT)

Formal parameters

X	Real array (N)	input:	sample
N	Integer	input:	sample size
MODE	Integer	input:	formulation; if zero, then Formulation B is used, else Formulation A
KSUM	Integer	input:	convolution power k
XLAM	Real	input:	λ
TAU	Real	input:	τ
NPTS	Integer	input:	q
TOL	Real	input:	convergence tolerance
MAXIT	Integer	input:	maximum allowable number of iterations
XL	Real array (NPAR)	input:	lower bounds for parameters $(\alpha,\beta,\sigma,\mu)$; the third and fourth elements change during execution
		output:	input values are restored
XH	Real array (NPAR)	input:	upper bounds for parameters; the third and fourth elements change during execution
		output:	input values are restored
NPAR	Integer	input:	number of parameters (4)
ISCLBD	Integer	input:	flag if $\tilde{\sigma}$ is constrained: if negative, $\tilde{\sigma}$ is fixed at XB(3); if zero, $\tilde{\sigma}$ is free to vary and initial values of XL(3) and XH(3) are irrelevant; if positive, $\tilde{\sigma}$ is constrained in [XL(3),XH(3)]

LOCBND	Integer	input:	flag if $\tilde{\mu}$ is constrained: if negative, $\tilde{\mu}$ is fixed at XB(4); if zero, $\tilde{\mu}$ is free to vary and initial values of XL(4) and XH(4) are irrelevant; if positive, $\tilde{\mu}$ is constrained in [XL(4),XH(4)]
ICOV	Integer	input:	flag for computation of covariance matrices: if negative, neither approximation (i) nor (ii) is computed; if zero, both approximations are computed; if positive, only approximation (ii) is computed
VCV1	Real array(NPAR,NPAR)	output:	covariance matrix approximation (i) if requested; the strict lower triangle contains correlations, the upper triangle contains covariances (times n); if a parameter is on a bound, the corresponding elements are zero
VCV2	Real array(NPAR,NPAR)	output:	covariance matrix approximation (ii) if requested; the strict lower triangle contains correlations, the upper triangle covariances (times n); if a parameter is on a bound, the corresponding elements are zero
WORK	Real array (LWORK)	workspace:	
		output:	some elements may be of interest on output (see Restrictions)
LWORK	Integer	input:	
IWORK	Integer array(LIWORK)	workspace:	
		output:	some elements may be of interest on output (see Restrictions)
LIWORK	Integer	input:	
IVNIT	Integer	input:	if positive, unit number for output (see Additional Comments); if zero or negative, no output is produced
IFAULT	Integer	output:	failure indicator

Failure indicators

IFAULT = 0 indicates success. Nonzero values of IFAULT are due to two types of errors. The first type is input errors, detected in STABLE; IFAULT will be

1 if MAXIT≤0;

2 if N<50 (see Restrictions);

3 if KSUM ≤ 0;

4 if τ<0 or τ>1 and MODE≠0;

5 if NPTS<20 or mod(NPTS,2)≠0;

6 if TOL≤0;

7 if NPAR≠4;

8 if insufficient workspace was allotted (see Restrictions);

9 if improper bounds were supplied. The following conditions cause this failure:
XL(i)>XB(i) or XL(i)>XH(i) or XB(i)>XH(i), i=1,2
XL(1)≤0 or XH(1)>2
XL(1)<-1 or XH(1)>1
(XL(2)≠0 or XH(2)≠0) and (XL(1)<1 and XH(1)≥1
or XL(1)≤1 and XH(1)>1)
XB(3)≤0
ISCLBD≠0 and (XL(3)>XB(3) or XL(3)>XH(3) or XB(3)>XH(3))
ISCLBD<0 and XL(3)≠XH(3)
LOCBND≠0 and (XL(4)>XB(4) or XL(4)>XH(4) or XB(4)>XH(4))
LOCBND<0 and XL(4)≠XH(4)
LOCBND≠0 and XL(1)=XH(1)=1 and (XL(2)≠0 or XH(2)≠0).

On input errors, STABLE terminates immediately, without performing any computations. The second type of error occurs after some computation. IFAULT will be

10 if A was found numerically non positive definite;

11 if A was found ill-conditioned;

12 if too many function evaluations were required during solution of a subproblem;

13 if iteration converged, but the most recent E04KBF fault indicator was 3 and internal checks were not met. These checks are
(i) $||G||^2 < 10 * $ X02AAF(DUMMY), and
(ii) $K < 1/||G||$, as recommended by E04KBF documentation, where $||G||$ is the norm of the projected gradient and K the estimated condition number of the projected Hessian matrix;

14 if there were repeated problems with overflow in the Cholesky factors of the projected Hessian;

15 if iteration converged, but the most recent E04KBF fault indicator was 5 and internal checks were not met;

16 if convergence did not occur in MAXIT iterations;

17 if convergence did not occur in MAXIT iterations, the most recent E04KBF fault indicator was 3, and internal checks were not met;

18 if convergence did not occur in MAXIT iterations, the most recent E04KBF fault indicator was 5, and internal checks were not met.

Conditions IFAULT=10 and 11 are detected in SETECF (they are caused by τ being too small under Formulation B), the remainder in STABLE.

IWORK(2) and IWORK(3) (see Restrictions) are failure indicators for asymptotic covariance matrix versions (i) and (ii) respectively. Zero indicates success, 1 that $\underset{\sim}{H}$ was non positive definite, and 2 that $\underset{\sim}{H}$ was ill-conditioned, the failures detected in SETVCV. If IFAULT=1-12 or 14, covariance matrices are not computed, and their fault indicators are set to the corresponding value of IFAULT.

Auxiliary algorithms

The user has only to call STABLE. Auxiliary procedures fall into two groups: those supplied here, and NAG Library procedures. The following subroutines are supplied:

SUBROUTINE GRIDWT(PAR,NPAR,XLAM,TAU,PTS,NPTS2,WT,NPTS,MODE,KSUM): computes gridpoints and weights;

SUBROUTINE CHARFN(U,PAR,NPAR,RE,XIM): computes real and imaginary parts of standard stable characteristic function $\tilde{\phi}(u)$;

SUBROUTINE FUNCT(IFLAG,N,XC,FC,GC,IW,LIW,W,LW): objective function/gradient evaluation;

SUBROUTINE SETECF(X,N,PAR,NPAR,MODE,TAU,SIGMA,XMU,KSUM,IA,NPTS2,NPTS, PTS,ECF,A,AINV,WORK,IFAULT): computes standardized empirical characteristic function values $\tilde{\psi}_n^k(u)$, computes and inverts A under Formulation B;

SUBROUTINE MONIT(N,XC,FC,GC,ISTATE,GPJNRM,COND,POSDEF,NITER,NF,IW,LIW, W,LW): monitors the progress of E04KBF;

SUBROUTINE VARIAB(ICOV,X,N,PAR,NPAR,MODE,SIGMA,XMU,ISUB,NVAR,PTS,NPTS2, WT,ECF,NPTS,DERIV,WORK,HOLD,A,IA,AINV,VCV1,VCV2,H,NVAR1,V,IW,LIW,W,LW, IFAIL1,IFAIL2): computes approximate asymptotic covariance matrices;

SUBROUTINE VMATRX (X,N,MODE,XMU,SIGMA,PTS,NPTS2,WT,ECF,WORK,NPTS,DERIV, V,HOLD,NVAR): computes $\underset{\sim}{V}$ for version (i) of asymptotic covariance matrix;

SUBROUTINE DAPROD(FAC1,IFAC1,NPTS,FAC2,WORK,NVAR): auxiliary matrix multiplication for VARIAB:

SUBROUTINE HVPROD(FAC1,IFAC1,NVAR,FAC2,NPTS,VH,IVH): auxiliary matrix multiplication for VARIAB;

SUBROUTINE SETVCV(ISUB,NVAR,H,NVAR1,V,WORK,VCV,NPAR,SIGMA,IFAULT): auxiliary routine for VARIAB;

SUBROUTINE HESDIF(PAR,NPAR,ISUB,H,SAVE1,SAVE2,NVAR,IW,LIW,W,LW): computes an approximate Hessian by differencing for version (i) of asymptotic covariance matrix.

The following NAG Library procedures are used:

REAL FUNCTION X02AAF(DUMMY): returns the smallest positive ϵ such that $1.0 + \epsilon > 1.0$;

SUBROUTINE E04KBF(N,FUNCT,MONIT,IPRINT,LOCSCH,INTYPE,MINLIN,MAXCAL,ETA, XTOL,STEPMX,FEST,IBOUND,BL,BU,X,HESL,LH,HESD,ISTATE,F,G,IW,LIW,W,LW,IFAIL): solves optimization problems. Control parameters are set as follows:

IPRINT	=	0
LOCSCH	=	.TRUE.
INTYPE	=	3 for subproblems after the first if parameters which are not fixed are not on bounds, else 0
MINLIN	=	NAG Library routine E04LBS
MAXCAL	=	400
ETA	=	0.9
XTOL	=	$10.0\sqrt{\text{X02AAF(DUMMY)}}$ explicitly, so it is available on exit
STEPMX	=	0.25
FEST	=	0.0
IBOUND	=	0 ;

SUBROUTINE F01ABF(A,IA,N,B,IB,Z,IFAIL): inverts the positive definite symmetric matrix A;

SUBROUTINE F01CAF(A,M,N,IFAIL): sets matrix A to zero;

SUBROUTINE F01CMF(A,LA,B,LB,M,N): copies elements of matrix A into matrix B;

SUBROUTINE F01CKF(A,B,C,N,IP,M,Z,IZ,IOPT,IFAIL): matrix multiplication A=BC, where B or C may be overwritten.

RESTRICTIONS

We require the sample size N at least 50, since for smaller samples $\tilde{\psi}_n(u)$ is not generally sufficiently smooth to allow accurate estimation. Since $\tilde{\alpha}$ and $\tilde{\beta}$ are bounded in the narrow ranges (0,2] and [-1,1] and have standard errors decreasing as $N^{-\frac{1}{2}}$, it is preferable to have N≥100. For N less than 150, say, relatively large values of λ may be preferred, to damp out noise in $\tilde{\psi}_n(u)$. We further require NPTS≥20.

Extended work vectors WORK and IWORK are required, in order to communicate information to FUNCT and MONIT without using COMMON blocks. To aid readers who may wish to adapt the algorithm to installations not having the NAG Library, we describe the use of these work vectors.

The required length of WORK is 10 + 11*NPAR + NPAR*(NPAR-1)/2 + (3+NPAR)*NPTS + NPTS + NPTS/2 if MODE≠0, with an additional NPTS*(2*NPTS+1) required if MODE=0. Some sample lengths are

MODE	NPTS=20	NPTS=40
0	1030	3600
nonzero	210	360

The subvector W is passed to E04KBF,FUNCT, and MONIT.

WORK starting point	W starting point	Elements	Used for
1	-	1	Convergence criterion
2	-	1	Objective function value on exit from E04KBF
3	-	NPAR	Projected gradient on exit from E04KBF
(other addresses internally computed)	-	1	Old $\tilde{\alpha}$ value for convergence testing
	-	1	Old $\tilde{\beta}$ value for convergence testing
	-	1	$\tilde{\sigma}$ lower bound on entry
	-	1	$\tilde{\sigma}$ upper bound on entry
	-	1	$\tilde{\mu}$ lower bound on entry
	-	1	$\tilde{\mu}$ upper bound on entry
	-	NPAR*(NPAR-1)/2	HESL factor for E04KBF
	-	NPAR	HESD factor for E04KBF; workspace for VARIAB
	1 (other addresses internally computed)	9*NPAR	E04KBF workspace; broken up into H and V matrices and used as workspace in VARIAB
		1	On exit from E04KBF, estimated condition number of projected Hessian
		1	On exit from E04KBF, norm of projected gradient
		NPTS/2	Positive gridpoints PTS
		NPTS	Weights WT
		NPTS	Empirical characteristic function values ECF, workspace in VARIAB
		NPAR*NPTS	Partial derivatives of $\psi(u)$ at gridpoints, workspace in VARIAB
		NPTS	Workspace for FUNCT and VARIAB
		(NPTS+1)*NPTS	If MODE=0, used for A matrix (the extra row is required by NAG Library routine F01ABF); workspace in VARIAB
		NPTS*NPTS	If MODE=0, used for A^{-1}; workspace in VARIAB

The required length of IWORK is 7 + NPAR. The subvector IW is passed to E04KBF, FUNCT, and MONIT.

IWORK starting point	IW starting point	Elements	Use for
1	-	1	Iteration count
2	-	NPAR	ISTATE vector for E04KBF, workspace for VARIAB.
(other addresses internally computed)			If covariance matrices are requested, on exit IWORK(2) contains a fault indicator for approximation (i), IWORK(3) contains a fault indicator for approximation (ii), and IWORK(4) contains the number of iterations required to compute the approximate Hessian for approximation (i)
	1	2	Workspace for E04KBF, HESDIF
	3	1	Stores MODE
	4	1	Stores output unit number IUNIT
	5	1	Stores NPTS
	6	1	Stores 1 less than the address of PTS(1) in W

PRECISION

Double precision will be required on computers with 32 bit wordlength. The precision used by the local NAG Library implementation should be adequate. To change the precision:

- change all REAL declarations to DOUBLE PRECISION;
- replace constants by double precision versions, constants $\frac{\pi}{2}$, $\frac{2}{\pi}$, $\sqrt{10}$ typed in to machine accuracy;
- declare NAG Library function X02AAF as DOUBLE PRECISION;
- change the precision of FORTRAN library functions, i.e., ABS to DABS, ATAN2 to DATAN2, SIGN to DSIGN, etc. FLOAT(I) can be replaced by DBLE(FLOAT(I)).

If extremely large observations are present in the sample, there may be a loss of significant figures when computing sines and cosines in SETECF and VMATRX. This should not occur when real data is used, but can be a problem with simulated data for small α.

TIME

Execution times depend on the quality of initial guesses and properties of the real data used, and vary somewhat throughout the parameter space. As a rough guide, we give some statistics for simulated data, using a moderately difficult situation with $\alpha>1$. Tables 1a and 1b provide approximate running times for Formulation A, q=40, and Formulation B, q=20, n=100,200,500,1000,2500. Timing starts upon entry to STABLE. Samples from S(1.3,-.5,3,15) were generated using the method of Chambers, Mallows, and Stuck (1976). Initial guesses for α,β,σ,μ in all cases were $1.505=\frac{1}{2}(1.01+2)$, 0, $\frac{1}{2}(x_{.75}-x_{.25})$, and $x_{.5}$, the sample median, respectively. Because of skewness, the median is not a good estimator of μ in this case. Five iterations were used. Time required to compute asymptotic covariance matrices includes approximations (i) and (ii), except where noted. Timings are for a double precision version of the algorithm, compiled by the IBM FORTRAN H Extended compiler, and run on an IBM 370/3033.

The following qualitative points are clear from this rather restricted set of timings. There is a substantial overhead, which may crudely be assumed fixed, associated with nonlinear optimization, although E04KBF solves the optimization subproblems rapidly. For large samples, run time is dominated by evaluation of the empirical characteristic function, and thus is asymptotically linear in n for a fixed number of iterations.

Table 1a

Timings for Formulation A, q=40, on Simulated Samples from s(1.3,-.5,3,15); λ=1 for n$\le$200 and .5 for n>200, τ=1

n	Iterations	Estimation time (sec)	Convergence criterion	Covariance matrix time
100	5	0.7	5.4(-4)	0.1*
200	5	1.0	2.3(-2)	0.1*
500	5	1.8	1.8(-4)	0.8
1000	5	3.2	3.3(-5)	1.2
2500	5	7.5	4.8(-6)	2.5

*Sample size too small to compute approximation (i), only approximation (ii) computed.

Table 1b

Timings for Formulation B, q=20, on Simulated Samples from s(1.3,-.5,3,15); λ=τ=0

n	Iterations	Estimation time (sec)	Convergence criterion	Covariance matrix time
100	5	0.9	1.2(-2)	0.3
200	5	1.2	3.2(-2)	0.4
500	5	1.6	1.8(-4)	0.5
1000	5	2.3	5.7(-5)	0.7
2500	5	4.4	1.5(-4)	1.4

Approximation (ii) of the asymptotic covariance matrix is quite easy to compute, while approximation (i) is highly time-consuming.

For fixed k>1 with the k-sum procedure, one iteration generally suffices, provided estimates from the nearest value of k are used, and the estimates don't change much. For mixtures of very different distributions, or if the exponent $\tilde{\alpha}$ is near unity, more are required.

ADDITIONAL COMMENTS

Although output need not be produced, we recommend calling STABLE with IUNIT>0, so the user will have a record of how estimation progressed. The following information will then be printed out:

by MONIT: number of E04KBF iterations and function evaluations, objective function value, norm of projected gradient, subproblem solution, projected gradient, and estimated condition number of projected Hessian;

by STABLE: E04KBF fault indicator, and value of convergence criterion;

by HESDIF(if called): number of iterations needed to compute approximate Hessian, and steplength used.

Use of STABLE in "batch mode" has drawbacks. For instance, most faults arising in E04KBF are not diagnosed until iteration ceases. In practice, such faults may likely be due to the initial $\tilde{\alpha}$ being on the wrong side of 1. Further, when $\tilde{\alpha}$ is small, convergence tolerances are difficult to interpret, and the user may prefer direct control of iteration. We therefore prefer to use STABLE interactively, a copy of the output described above being directed to the terminal, and the user deciding after each iteration whether he wishes to continue. Required modifications are simple.

Faster and/or more compact codings of the Algorithm are possible, for instance, if β is known to be zero, if only Formulation A or Formulation B is desired, or if asymptotic covariance matrices are not desired. Generality is achieved at a price in efficiency.

REFERENCES

Brockwell, P.J. and Brown, B.M. (1981). High-efficiency estimation for the positive stable laws. J. Amer. Statist. Assoc., 76, 626-631.

Chambers, J.M. (1977). Computational Methods for Data Analysis. New York: Wiley.

Chambers, J.M., Mallows, C.L. and Stuck, B.W. (1976). A method for simulating stable random variables. J. Amer. Statist. Assoc., 71, 340-344.

DuMouchel, W.H. (1971). Stable distributions in statistical inference. Unpublished Ph.D. thesis, Department of Statistics, Yale University.

Fama, E.F. and Roll, R. (1971). Parameter estimation for symmetric stable distributions. J. Amer. Statist. Assoc., 66, 331-338.

Feuerverger, A. and McDunnough, P. (1981a). On the efficiency of empirical characteristic function procedures. J. R. Statist. Soc. B, 43, 20-27.

Feuerverger, A. and McDunnough, P. (1981b). On some Fourier methods for inference. J. Amer. Statist. Assoc., 76, 379-387.

Hiebert, K.L. (1981). An evaluation of mathematical software that solves nonlinear least squares problems. ACM Transactions on Mathematical Software, 7, 1-16.

Koutrouvelis, I.A. (1980). Regression-type estimation of the parameters of stable laws. J. Amer. Statist. Assoc., 75, 918-928.

Koutrouvelis, I.A. (1981). An iterative procedure for the estimation of the parameters of stable laws. Communications in Statistics B, 10, 17-28.

Mandelbrot, B. (1963). The variation of certain speculative prices. Journal of Business, 36, 394-419.

NAG FORTRAN Library Manual, Mark 8, The Numerical Algorithms Group (USA) Inc., Downers Grove, ILL., 1981.

Paulson, A.S., Holcomb, E.W. and Leitch, R.A. (1975). The estimation of the parameters of the stable laws. Biometrika, 62, 163-170.

Paulson, A.S., Delehanty, T.A. and Brothers, K.M. (1982). Some properties of modified integrated squared error estimators for the stable laws. Submitted for publication.

Paulson, A.S., and Delehanty, T.A. (1982). Modified weighted squared error estimation procedures with special emphasis on the stable laws. Submitted for publication.

```
C*********************************************************************STAB 001
C        CENTRAL SUBROUTINE OF ESTIMATION PROCESS.                    STAB 002
C        CALLED BY - MAIN                                             STAB 003
C        CALLS - GRIDWT, SETECF, VARIAB                               STAB 004
C        PASSES TO E04KBF VIA EXTERNAL STMT - FUNCT, MONIT, E04LBS    STAB 005
C        N.A.G. PROCEDURES CALLED -                                   STAB 006
C               X02AAF (RETURNS MACHINE PRECISION),                   STAB 007
C               E04KBF (QUASI-NEWTON OPTIMIZATION WITH BOUNDED        STAB 008
C                      VARIABLES),                                    STAB 009
C               F01CAF (SETS A MATRIX TO ZERO).                       STAB 010
C*********************************************************************STAB 011
      SUBROUTINE STABLE(X, N, MODE, KSUM, XLAM, TAU, NPTS, TOL, MAXIT,  STAB 012
     *XL, XB, XH, NPAR, ISCLBD, LOCBND, ICOV, VCV1, VCV2, WORK, LWORK,  STAB 013
     *IWORK, LIWORK, IUNIT, IFAULT)                                     STAB 014
C        EXTERNAL ROUTINES                                            STAB 015
      EXTERNAL E04LBS, FUNCT, MONIT                                   STAB 016
C        ARGUMENTS                                                    STAB 017
      INTEGER N, MODE, KSUM, NPTS, MAXIT, NPAR, ISCLBD, LOCBND, ICOV,   STAB 018
     *LWORK, LIWORK, IWORK(LIWORK), IUNIT, IFAULT                       STAB 019
      REAL X(N), XLAM, TAU, TOL, XL(NPAR), XB(NPAR), XH(NPAR),          STAB 020
     *VCV1(NPAR,NPAR), VCV2(NPAR,NPAR), WORK(LWORK)                     STAB 021
C        FUNCTION CALLED                                              STAB 022
      REAL X02AAF                                                     STAB 023
C        LOCAL SCALARS                                                STAB 024
C        SPECIFICALLY FOR E04KBF PARAMETERS -                         STAB 025
      LOGICAL LOCSCH                                                  STAB 026
      INTEGER LW, LIW, MAXCAL, IBOUND, IPRINT, LHESL, INTYPE          STAB 027
      REAL ETA, XTOL, STEPMX, FEST                                    STAB 028
C        SCRATCH VARIABLES, SUBSCRIPT INDICATORS, AND CONSTANTS -     STAB 029
      INTEGER IND, IND1, IOVFLW, NPTS2, IA, NVAR, NVAR1, MPTS, MWT,   STAB 030
     *MECF, MDERIV, MVWORK, MA, MAINV, MHESL, MHESD, MIW, MW, MV       STAB 031
      REAL SIGMA, XMU, XK, ZERO, PT2, TWOVPI, ONE, TWO, SQRT10, FIVE,  STAB 032
     *TEN                                                             STAB 033
      DATA LOCSCH, MAXCAL, IBOUND, IPRINT, ETA, STEPMX, FEST          STAB 034
     */.TRUE., 400, 0, 0, 0.9, 0.25, 0.0/                              STAB 035
      DATA ZERO, PT2, TWOVPI, ONE, TWO, SQRT10, FIVE, TEN             STAB 036
     */0.0, 0.2, 0.6366197724, 1.0, 2.0, 3.162277660, 5.0, 10.0/       STAB 037
C                                                                     STAB 038
C        ON INPUT ERRORS (IFAULT = 1 - 9), EXIT IMMEDIATELY.          STAB 039
      IWORK(1) = 0                                                    STAB 040
      WORK(1) = ZERO                                                  STAB 041
      IFAULT = 1                                                      STAB 042
      IF (MAXIT .LE. 0) RETURN                                        STAB 043
      IFAULT = 2                                                      STAB 044
      IF (N .LT. 50) RETURN                                           STAB 045
      IFAULT = 3                                                      STAB 046
      IF (KSUM .LE. 0) RETURN                                         STAB 047
      IFAULT = 4                                                      STAB 048
      IF (TAU .LT. ZERO .OR. MODE .NE. 0 .AND. TAU .GT. ONE) RETURN   STAB 049
      IFAULT = 5                                                      STAB 050
      IF (NPTS .LT. 20 .OR. MOD(NPTS,2) .NE. 0) RETURN                STAB 051
      IFAULT = 6                                                      STAB 052
      IF (TOL .LE. ZERO) RETURN                                       STAB 053
      IFAULT = 7                                                      STAB 054
      IF (NPAR .NE. 4) RETURN                                         STAB 055
      IFAULT = 8                                                      STAB 056
      NPTS2 = NPTS / 2                                                STAB 057
      LHESL = (NPAR*NPAR - NPAR) / 2                                  STAB 058
      LW = 10 + 11 * NPAR + LHESL + (3 + NPAR) * NPTS + NPTS2         STAB 059
      IF (MODE .EQ. 0) LW = LW + NPTS * (2*NPTS + 1)                  STAB 060
```

```
      IF (LWORK .LT. LW .OR. LIWORK .LT. NPAR + 7) RETURN               STAB 061
C                                                                       STAB 062
C        CHECKING OF PARAMETER BOUNDS (IFAULT = 9 IF WRONG)             STAB 063
      IFAULT = 9                                                        STAB 064
      IF (XL(1) .GT. XB(1) .OR. XL(1) .GT. XH(1) .OR. XB(1) .GT. XH(1)) STAB 065
     *RETURN                                                            STAB 066
      IF (XL(1) .LE. ZERO .OR. XH(1) .GT. TWO) RETURN                   STAB 067
      IF (XL(2) .GT. XB(2) .OR. XL(2) .GT. XH(2) .OR. XB(2) .GT. XH(2)) STAB 068
     *RETURN                                                            STAB 069
      IF (XL(2) .LT. - ONE .OR. XH(2) .GT. ONE) RETURN                  STAB 070
      IF ((XL(2) .NE. ZERO .OR. XH(2) .NE. ZERO) .AND. (XL(1) .LT. ONE  STAB 071
     *.AND. XH(1) .GE. ONE .OR. XL(1) .LE. ONE .AND. XH(1) .GT. ONE))   STAB 072
     *RETURN                                                            STAB 073
      IF (XB(3) .LE. ZERO) RETURN                                       STAB 074
      IF (ISCLBD .NE. 0 .AND. (XL(3) .GT. XB(3) .OR. XL(3) .GT. XH(3) . STAB 075
     *OR. XB(3) .GT. XH(3))) RETURN                                     STAB 076
      IF (ISCLBD .LT. 0 .AND. XL(3) .NE. XH(3)) RETURN                  STAB 077
      IF (LOCBND .NE. 0 .AND. (XL(4) .GT. XB(4) .OR. XL(4) .GT. XH(4) . STAB 078
     *OR. XB(4) .GT. XH(4))) RETURN                                     STAB 079
      IF (LOCBND .LT. 0 .AND. XL(4) .NE. XH(4)) RETURN                  STAB 080
      IF (LOCBND .NE. 0 .AND. XL(1) .EQ. ONE .AND. XH(1) .EQ. ONE .AND. STAB 081
     *(XL(2) .NE. ZERO .OR. XL(2) .NE. XH(2))) RETURN                   STAB 082
C                                                                       STAB 083
C        INITIAL ADJUSTMENT OF LOCATION/SCALE PARAMETERS/BOUNDS.        STAB 084
C        SAVE BOUNDS FOR EXIT.                                          STAB 085
      WORK(NPAR + 5) = XL(3)                                            STAB 086
      WORK(NPAR + 6) = XH(3)                                            STAB 087
      WORK(NPAR + 7) = XL(4)                                            STAB 088
      WORK(NPAR + 8) = XH(4)                                            STAB 089
      XK = FLOAT(KSUM)                                                  STAB 090
C        LOCATION                                                       STAB 091
      XMU = XK * XB(4)                                                  STAB 092
      IF (LOCBND .LT. 0) GO TO 10                                       STAB 093
      IF (LOCBND .GT. 0) GO TO 20                                       STAB 094
      XL(4) = -FIVE                                                     STAB 095
      XH(4) = FIVE                                                      STAB 096
      GO TO 30                                                          STAB 097
   10 XL(4) = ZERO                                                      STAB 098
      XH(4) = ZERO                                                      STAB 099
      GO TO 30                                                          STAB 100
   20 XL(4) = XK * XL(4)                                                STAB 101
      XH(4) = XK * XH(4)                                                STAB 102
C        SCALE                                                          STAB 103
   30 XTOL = XK ** (ONE/XB(1))                                          STAB 104
      SIGMA = XTOL * XB(3)                                              STAB 105
      IF (ISCLBD .LT. 0) GO TO 40                                       STAB 106
      IF (ISCLBD .GT. 0) GO TO 50                                       STAB 107
      XL(3) = PT2                                                       STAB 108
      XH(3) = FIVE                                                      STAB 109
      GO TO 60                                                          STAB 110
   40 XL(3) = ONE                                                       STAB 111
      XH(3) = ONE                                                       STAB 112
      GO TO 60                                                          STAB 113
   50 XL(3) = XTOL * XL(3)                                              STAB 114
      XH(3) = XTOL * XH(3)                                              STAB 115
C                                                                       STAB 116
C        EXPLICITLY SET XTOL TO EO4KBF DEFAULT VALUE, USING XO2AAF,     STAB 117
C        SO THAT IT IS AVAILABLE ON EXIT ON EO4KBF.                     STAB 118
   60 XTOL = TEN * SQRT(XO2AAF(XTOL))                                   STAB 119
C                                                                       STAB 120
```

```
C        MEMORY MANAGEMENT
      MIW = 2 + NPAR
      IWORK(MIW + 2) = MODE
      IWORK(MIW + 3) = IUNIT
      IWORK(MIW + 4) = NPTS
      IWORK(MIW + 5) = 9 * NPAR + 2
      LIW = 6
      LW = LW - 8 - 2 * NPAR - LHESL
      MHESL = 9 + NPAR
      MHESD = MHESL + LHESL
      MW = MHESD + NPAR
      MPTS = MW + 9 * NPAR + 2
      MWT = MPTS + NPTS2
      MECF = MWT + NPTS
      MDERIV = MECF + NPTS
C        AVOID UNCLEAR REFERENCES TO A AND A INVERSE IF MODE .NE. 0
      MA = MDERIV
      MAINV = MDERIV
      IA = NPTS + 1
      IF (MODE .NE. 0) GO TO 70
      MA = MA + (NPAR + 1) * NPTS
      MAINV = MA + IA * NPTS
C        LOOPING POINT FOR ITERATION
   70 IWORK(1) = IWORK(1) + 1
      IOVFLW = 0
C
C        IF LOCATION/SCALE PARAMETERS ARE CONSTRAINED, UPDATE
C        THEIR BOUNDS.
      IF (ISCLBD .LE. 0) GO TO 80
      XL(3) = AMIN1(ONE,AMAX1(PT2,WORK(NPAR + 5)/SIGMA))
      XH(3) = AMAX1(ONE,AMIN1(FIVE,WORK(NPAR + 6)/SIGMA))
   80 IF (LOCBND .LE. 0) GO TO 90
      XL(4) = AMIN1(ZERO,AMAX1(-FIVE,(WORK(NPAR + 7) - XMU)/SIGMA))
      XH(4) = AMAX1(ZERO,AMIN1(FIVE,(WORK(NPAR + 8) - XMU)/SIGMA))
C        CHANGE PARAMETERS TO REFLECT FUTURE STANDARDIZATION.
   90 XB(3) = ONE
      XB(4) = ZERO
      WORK(NPAR + 3) = XB(1)
      WORK(NPAR + 4) = XB(2)
C        SET INTYPE = 3 IF POSSIBLE, SO OLD HESSIAN CAN BE USED.
      INTYPE = 0
      IF (IWORK(1) .EQ. 1) GO TO 110
      INTYPE = 3
      DO 100 I = 1, NPAR
      IF (IWORK(I + 1) .GT. 0) GO TO 100
      INTYPE = 0
      GO TO 110
  100 CONTINUE
C
C        SET GRIDPOINTS, WEIGHTS.
  110 CALL GRIDWT(XB, NPAR, XLAM, TAU, WORK(MPTS), NPTS2, WORK(MWT),
     *NPTS, MODE, KSUM)
C        SET E. CH. F. VALUES, ALSO A AND A INVERSE, IF MODE = 0, USING
C        FIRST COL. OF DERIV AS WORKSPACE.
      CALL SETECF(X, N, XB, NPAR, MODE, TAU, SIGMA, XMU, KSUM, IA,
     *NPTS2, NPTS, WORK(MPTS), WORK(MECF), WORK(MA), WORK(MAINV),
     *WORK(MDERIV), IFAULT)
      IF (IFAULT .GT. 0) IFAULT = 9 + IFAULT
      IF (IFAULT .EQ. 10 .OR. IFAULT .EQ. 11) GO TO 180
C
```

```
C        CALL OR RESTART N.A.G. ROUTINE E04KBF                          STAB 181
  120 IFAULT = 1                                                        STAB 182
      CALL E04KBF(NPAR, FUNCT, MONIT, IPRINT, LOCSCH, INTYPE, E04LBS,   STAB 183
     *MAXCAL, ETA, XTOL, STEPMX, FEST, IBOUND, XL, XH, XB, WORK(MHESL), STAB 184
     *LHESL, WORK(MHESD), IWORK(2), WORK(2), WORK(3), IWORK(MIW), LIW,  STAB 185
     *WORK(MW), LW, IFAULT)                                             STAB 186
C        COMPUTE CONVERGENCE CRITERION                                  STAB 187
      WORK(1) = AMAX1(ABS(XB(3) - ONE),ABS(XB(4)))                      STAB 188
      IF (ISCLBD .LT. 0 .AND. LOCBND .LT. 0) WORK(1) = AMAX1(ABS(XB(1) -STAB 189
     * WORK(NPAR + 3)),ABS(XB(2) - WORK(NPAR + 4)))                     STAB 190
C        OUTPUT IF REQUESTED                                            STAB 191
      IF (IUNIT .GT. 0) WRITE (IUNIT,1000) IWORK(1), IFAULT, WORK(1)    STAB 192
      IF (IFAULT .EQ. 0) GO TO 140                                      STAB 193
C        IF E04KBF FAULT INDICATOR IS 2, 3, 4, OR 5, JUDGE QUALITY      STAB 194
C        OF SOLUTION.  NOTE IFAULT = 1 IS IMPOSSIBLE.                   STAB 195
      IFAULT = 10 + IFAULT                                              STAB 196
C        IF MAXCAL FUNCTION EVALUATIONS HAVE BEEN MADE, GIVE UP         STAB 197
      IF (IFAULT .EQ. 12) GO TO 180                                     STAB 198
      IF (IFAULT .NE. 14) GO TO 130                                     STAB 199
C        ONE OVERFLOW IN HESSIAN CHOLESKY FACTORS IS ALLOWED            STAB 200
      IF (IOVFLW .GE. 1) GO TO 180                                      STAB 201
      IOVFLW = IOVFLW + 1                                               STAB 202
      INTYPE = 0                                                        STAB 203
      GO TO 120                                                         STAB 204
C        IF IFAULT = 13 OR 15, TEST PROJECTED GRADIENT AND PROJECTED    STAB 205
C        HESSIAN CONDITION NUMBER                                       STAB 206
  130 IND = MW + 9 * NPAR                                               STAB 207
      IF (SQRT10*WORK(IND + 1) .LT. XTOL .AND. WORK(IND) .LT. ONE/WORK( STAB 208
     *IND + 1)) IFAULT = 0                                              STAB 209
C                                                                       STAB 210
C        UPDATE LOCATION AND SCALE AND TEST CONVERGENCE                 STAB 211
  140 XMU = XMU + SIGMA * XB(4)                                         STAB 212
      IF (XB(1) .EQ. ONE) XMU = XMU + SIGMA * TWOVPI * XB(2) * XB(3) *  STAB 213
     *ALOG(XB(3))                                                       STAB 214
      SIGMA = SIGMA * XB(3)                                             STAB 215
      IF (WORK(1) .LT. TOL) GO TO 150                                   STAB 216
      IF (IWORK(1) .LT. MAXIT) GO TO 70                                 STAB 217
C                                                                       STAB 218
C        MAXIT ITNS HAVE BEEN USED WITHOUT CONVERGENCE - SET IFAULT.    STAB 219
      IND = 16                                                          STAB 220
      IF (IFAULT .EQ. 13) IND = 17                                      STAB 221
      IF (IFAULT .EQ. 15) IND = 18                                      STAB 222
      IFAULT = IND                                                      STAB 223
C                                                                       STAB 224
C        TERMINATION WITH IFAULT = 0, 13, 15, 16, 17 OR 18              STAB 225
  150 IF (KSUM .GT. 1 .OR. ICOV .LT. 0) GO TO 190                       STAB 226
C                                                                       STAB 227
C        PREPARE TO CALL VARIAB TO COMPUTE COVARIANCE MATRICES.         STAB 228
C        MEMORY MANAGEMENT.                                             STAB 229
      MVWORK = MDERIV + NPAR * NPTS                                     STAB 230
      MV = MW + NPAR * (NPAR + 1)                                       STAB 231
C                                                                       STAB 232
C        REARRANGE THE N.A.G. ISTATE VECTOR SO IT HOLDS FREE            STAB 233
C        PARAMETER SUBSCRIPTS IN INCREASING ORDER.                      STAB 234
      NVAR = 0                                                          STAB 235
      DO 160 I = 1, NPAR                                                STAB 236
      IF (IWORK(I + 1) .LT. 0) GO TO 160                                STAB 237
      NVAR = NVAR + 1                                                   STAB 238
      IWORK(NVAR + 1) = I                                               STAB 239
  160 CONTINUE                                                          STAB 240
```

```
C        IN THE RARE EVENT THAT THERE ARE NO FREE VARIABLES, VARIAB      STAB 241
C        IS NOT CALLED AND COVARIANCE MATRICES ARE SET TO ZERO.          STAB 242
      IF (NVAR .GT. 0) GO TO 170                                         STAB 243
      IF (ICOV .EQ. 0) CALL F01CAF(VCV1, NPAR, NPAR, IWORK(2))           STAB 244
      CALL F01CAF(VCV2, NPAR, NPAR, IWORK(3))                            STAB 245
      GO TO 190                                                          STAB 246
  170 NVAR1 = NVAR + 1                                                   STAB 247
      XB(3) = ONE                                                        STAB 248
      XB(4) = ZERO                                                       STAB 249
C                                                                        STAB 250
C        COMPUTE ESTIMATED ASYMPTOTIC COVARIANCE MATRICES.               STAB 251
      CALL VARIAB(ICOV, X, N, XB, NPAR, MODE, SIGMA, XMU, IWORK(2),      STAB 252
     *NVAR, WORK(MPTS), NPTS2, WORK(MWT), WORK(MECF), NPTS,              STAB 253
     *WORK(MDERIV), WORK(MVWORK), WORK(MHESD), WORK(MA), IA,             STAB 254
     *WORK(MAINV), VCV1, VCV2, WORK(MW), NVAR1, WORK(MV), IWORK(MIW),    STAB 255
     *LIW, WORK(MW), LW, IND, IND1)                                      STAB 256
C        SAVE FAULT INDICATORS AND NO. OF ITNS TAKEN FOR HESSIAN.        STAB 257
      IWORK(2) = IND                                                     STAB 258
      IWORK(3) = IND1                                                    STAB 259
      IWORK(4) = IWORK(MIW)                                              STAB 260
      GO TO 190                                                          STAB 261
C                                                                        STAB 262
C        EXIT FOR IFAULT = 10, 11, 12, OR 14.  IF IFAULT = 12 OR 14,     STAB 263
C        RESULTS OF THE LAST (ABORTED) OPTIMIZATION ARE NOT USED.        STAB 264
  180 IWORK(2) = IFAULT                                                  STAB 265
      IWORK(3) = IFAULT                                                  STAB 266
C                                                                        STAB 267
C        EXIT FOR IFAULT = 0, 13, 15, 16, 17, OR 18.                     STAB 268
C        RESET LOCATION/SCALE BOUNDS                                     STAB 269
  190 XL(3) = WORK(NPAR + 5)                                             STAB 270
      XH(3) = WORK(NPAR + 6)                                             STAB 271
      XL(4) = WORK(NPAR + 7)                                             STAB 272
      XH(4) = WORK(NPAR + 8)                                             STAB 273
C        ADJUST PARAMETERS TO STANDARD FORM                              STAB 274
      IF (XB(1) .EQ. ONE) XMU = XMU - TWOVPI * XB(2) * SIGMA * ALOG(     STAB 275
     *SIGMA)                                                             STAB 276
      XB(4) = XMU / XK                                                   STAB 277
      XB(3) = SIGMA / (XK**(ONE/XB(1)))                                  STAB 278
C                                                                        STAB 279
 1000 FORMAT (10HOMAJOR ITN, I3, 18H, IFAIL (E04KBF) =, I3,              STAB 280
     *25H, CONVERGENCE CRITERION =, E11.3)                               STAB 281
      RETURN                                                             STAB 282
      END                                                                STAB 283
C***********************************************************************GRID 001
C        CALCULATES (POSITIVE) GRIDPOINTS AND ALL WEIGHTS.               GRID 002
C        CALLED BY - STABLE                                              GRID 003
C        CALLS - CHARFN                                                  GRID 004
C***********************************************************************GRID 005
      SUBROUTINE GRIDWT(PAR, NPAR, XLAM, TAU, PTS, NPTS2, WT, NPTS,      GRID 006
     *MODE, KSUM)                                                        GRID 007
C        ARGUMENTS                                                       GRID 008
      INTEGER NPAR, NPTS2, NPTS, MODE, KSUM                              GRID 009
      REAL PAR(NPAR), XLAM, TAU, PTS(NPTS2), WT(NPTS)                    GRID 010
C        LOCAL SCALARS                                                   GRID 011
      LOGICAL FLAG                                                       GRID 012
      INTEGER IND, IND1, IND2, INNER                                     GRID 013
      REAL ALPHA, AFAC1, AFAC2, TEMP, GAP, END, C1                       GRID 014
C        CONSTANTS                                                       GRID 015
      REAL ZERO, PT025, PT035, PT04, PT0425, PT045, PT05, PT06, PT07,    GRID 016
     *PT075, PT3, PT5, PT6, PT8, PT9, ONE, ONEPT2, ONEPT4, ONEPT6,       GRID 017
```

```
     *ONEPT7, ONEPT8, TWO, THREE, THRPT3, THRPT6, FOUR, FIVE, SEVEN, TEN
      DATA ZERO, PT025, PT035, PT04, PT0425, PT045, PT05, PT06, PT07,
     *PT075, PT3, PT5, PT6, PT8, PT9, ONE, ONEPT2, ONEPT4, ONEPT6,
     *ONEPT7, ONEPT8, TWO, THREE, THRPT3, THRPT6, FOUR, FIVE, SEVEN,
     *TEN /0.0, 0.025, 0.035, 0.04, 0.0425, 0.045, 0.05, 0.06, 0.07,
     *0.075, 0.3, 0.5, 0.6, 0.8, 0.9, 1.0, 1.2, 1.4, 1.6, 1.7, 1.8, 2.0,
     *3.0, 3.3, 3.6, 4.0, 5.0, 7.0, 10.0/
C
C         SELECT NUMBER OF INNER GRIDPOINTS.
C         NUMBER OF INNER PTS = 2 IF NPTS .LT. 30, 3 IF NPTS .GE. 30.
      INNER = 2
      IF (NPTS .GE. 30) INNER = 3
      ALPHA = PAR(1)
      IF (ALPHA .LE. ONE) GO TO 70
C
C         CASE WHEN ALPHA .GT. 1 - FIRST CHOOSE RIGHT ENDPOINT.
      END = THREE
      IF (ALPHA .LT. ONEPT8) END = THRPT3
      IF (ALPHA .LT. ONEPT7) END = THRPT6
      IF (KSUM .EQ. 1) GO TO 20
C
C         WHEN KSUM = 1, EQUISPACE POINTS.
      TEMP = ZERO
      GAP = END / FLOAT(NPTS2)
      DO 10 I = 1, NPTS2
      TEMP = TEMP + GAP
      PTS(I) = TEMP
   10 CONTINUE
      GO TO 100
C
C         WHEN KSUM .GT. 1 - FIRST USE HALF OF F-M ALPHA OPTIMAL GAPS
C         FOR POINTS CLOSE TO ORIGIN.
   20 IF (NPTS .GE. 30) GO TO 30
      GAP = PT075
      IF (ALPHA .LT. ONEPT8) GAP = PT06
      IF (ALPHA .LT. ONEPT6) GAP = PT07
      IF (ALPHA .LT. ONEPT4) GAP = PT075
      IF (ALPHA .LT. ONEPT2) GAP = PT0425
      GO TO 40
   30 GAP = PT05
      IF (ALPHA .LT. ONEPT8) GAP = PT035
      IF (ALPHA .LT. ONEPT6) GAP = PT04
      IF (ALPHA .LT. ONEPT4) GAP = PT045
      IF (ALPHA .LT. ONEPT2) GAP = PT025
   40 TEMP = ZERO
      DO 50 I = 1, INNER
      TEMP = TEMP + GAP
      PTS(I) = TEMP
   50 CONTINUE
C
C         LOGARITHMICALLY SPACE THE REST OF THE POINTS BY LOG(1 + U / 2)
      TEMP = ALOG(ONE + TEMP/TWO)
      GAP = (ALOG(ONE + END/TWO) - TEMP) / FLOAT(NPTS2 - INNER)
      IND = INNER + 1
      DO 60 I = IND, NPTS2
      TEMP = TEMP + GAP
      PTS(I) = TWO * (EXP(TEMP) - ONE)
   60 CONTINUE
      GO TO 100
C
```

```
C        CASE WHEN ALPHA .LE. 1 - SUBTRACT 1 FROM NO. OF INNER PTS        GRID 078
C        IF ALPHA .LE. 1/2.  START BY EQUISPACING ALL POINTS FOR          GRID 079
C        LOG(U + (MAX(ALPHA,0.3)**3), BUT MULTIPLY INNER POINTS BY        GRID 080
C        (MAX(ALPHA,0.3))**2 / 4.  (THE LATTER MULTIPLICATION IS          GRID 081
C        OMITTED IF KSUM .GT. 1.)  THEN CONTINUE SPACING.                 GRID 082
   70 END = FIVE                                                          GRID 083
      IF (ALPHA .LT. ONE) END = FOUR                                      GRID 084
      IF (ALPHA .LT. PT9) END = FIVE                                      GRID 085
      IF (ALPHA .LT. PT8) END = SEVEN                                     GRID 086
      IF (ALPHA .LT. PT6) END = TEN                                       GRID 087
      AFAC1 = AMAX1(ALPHA,PT3)                                            GRID 088
      AFAC2 = ONE                                                         GRID 089
      IF (KSUM .EQ. 1) AFAC2 = AFAC1 * AFAC1 / FOUR                       GRID 090
      AFAC1 = AFAC1 * AFAC1 * AFAC1                                       GRID 091
      TEMP = ALOG(AFAC1)                                                  GRID 092
      C1 = ALOG(END + AFAC1)                                              GRID 093
      GAP = (C1 - TEMP) / FLOAT(NPTS2)                                    GRID 094
      IF (ALPHA .LE. PT5) INNER = INNER - 1                               GRID 095
      DO 80 I = 1, INNER                                                  GRID 096
      TEMP = TEMP + GAP                                                   GRID 097
      PTS(I) = (EXP(TEMP) - AFAC1) * AFAC2                                GRID 098
   80 CONTINUE                                                            GRID 099
      TEMP = ALOG(PTS(INNER) + AFAC1)                                     GRID 100
      GAP = (C1 - TEMP) / FLOAT(NPTS2 - INNER)                            GRID 101
      IND = INNER + 1                                                     GRID 102
      DO 90 I = IND, NPTS2                                                GRID 103
      TEMP = TEMP + GAP                                                   GRID 104
      PTS(I) = EXP(TEMP) - AFAC1                                          GRID 105
   90 CONTINUE                                                            GRID 106
C                                                                         GRID 107
C        COMPUTE WEIGHTS (ALL NPTS OF THEM).                              GRID 108
C        IF KSUM .GT. 1, ALL WEIGHTS ARE 1.                               GRID 109
  100 IF (KSUM .EQ. 1) GO TO 120                                          GRID 110
      DO 110 I = 1, NPTS                                                  GRID 111
  110 WT(I) = ONE                                                         GRID 112
      RETURN                                                              GRID 113
C                                                                         GRID 114
  120 FLAG = MODE .NE. 0                                                  GRID 115
      IND = NPTS2 + 1                                                     GRID 116
      DO 140 I = 1, NPTS2                                                 GRID 117
      TEMP = PTS(I)                                                       GRID 118
      IND1 = NPTS2 + I                                                    GRID 119
      IND2 = IND - I                                                      GRID 120
      GAP = EXP(-XLAM*TEMP**ALPHA)                                        GRID 121
      IF (FLAG) GO TO 130                                                 GRID 122
      WT(IND1) = GAP                                                      GRID 123
      WT(IND2) = GAP                                                      GRID 124
      GO TO 140                                                           GRID 125
  130 GAP = GAP * GAP                                                     GRID 126
      CALL CHARFN(TEMP, PAR, NPAR, END, AFAC1)                            GRID 127
      CALL CHARFN(TEMP + TEMP, PAR, NPAR, AFAC2, C1)                      GRID 128
      WT(IND1) = GAP / (ONE + TAU*(C1 - (END + AFAC1)**2))                GRID 129
      WT(IND2) = GAP / (ONE - TAU*(C1 + (END - AFAC1)**2))                GRID 130
  140 CONTINUE                                                            GRID 131
C                                                                         GRID 132
      RETURN                                                              GRID 133
      END                                                                 GRID 134
C***********************************************************************CHAR 001
C        COMPUTES REAL AND IMAGINARY PARTS OF STANDARD STABLE             CHAR 002
C        CHARACTERISTIC FUNCTION (SIGMA = 1, MU = 0).                     CHAR 003
```

```
C        CALLED BY - GRIDWT, SETECF, VARIAB
C***********************************************************************
      SUBROUTINE CHARFN(U, PAR, NPAR, RE, XIM)
C        ARGUMENTS
      INTEGER NPAR
      REAL U, PAR(NPAR), RE, XIM
C        LOCAL SCALARS
      REAL ALPHA, XMOD, ZERO, ONE, PIBY2
      DATA ZERO, ONE, PIBY2 /0.0, 1.0, 1.570796327/
C
      RE = ABS(U)
      ALPHA = PAR(1)
      IF (ALPHA .NE. ONE) GO TO 10
      XIM = ZERO
      IF (U .NE. ZERO) XIM = ALOG(RE) / PIBY2
      GO TO 20
   10 XIM = TAN(PIBY2*ALPHA)
   20 XMOD = RE ** ALPHA
      XIM = -PAR(2) * SIGN(XMOD,U) * XIM
      XMOD = EXP(-XMOD)
      RE = XMOD * COS(XIM)
      XIM = XMOD * SIN(XIM)
C
      RETURN
      END
C***********************************************************************
C        FUNCTION/DERIVATIVE EVALUATION.
C        CALLED BY - E04KBF, VARIAB, HESDIF
C***********************************************************************
      SUBROUTINE FUNCT(IFLAG, N, XC, FC, GC, IW, LIW, W, LW)
C        ARGUMENTS
      INTEGER IFLAG, N, LIW, IW(LIW), LW
      REAL XC(N), FC, GC(N), W(LW)
C        LOCAL SCALARS
      LOGICAL ALF1, LFLAG
      INTEGER ILOW, NPTS, NPTS2, IND, IND1, ISUB, ISUB1, IPTS, IWT,
     *IECF, IDERIV, IPSI
      REAL ALPHA, BSIGN, SIGMA, XMU, OMEGA, XMOD, PTSI, SINE, COSINE,
     *PISEC2, SUEXP, XLOGSU, FAC, Z, Z1, ZERO, ONE, PIBY2
      DATA ZERO, ONE, PIBY2 /0.0, 1.0, 1.570796327/
C
      ALPHA = XC(1)
      BSIGN = XC(2)
      SIGMA = XC(3)
      XMU = XC(4)
      ALF1 = ALPHA .EQ. ONE
      LFLAG = IFLAG .EQ. 0
C
C        VARIABLES TO AID ADDRESSING IN W VECTOR - IPTS = ONE LESS THAN
C        POSITION OF FIRST POSITIVE GRIDPOINT IN W VECTOR, ETC.
      NPTS = IW(5)
      NPTS2 = NPTS / 2
      IND = NPTS + 1
      ILOW = NPTS2 + 1
      IPTS = IW(6)
      IWT = IPTS + NPTS2
      IECF = IWT + NPTS
      IDERIV = IECF + NPTS
      IPSI = IDERIV + N * NPTS
      NPTS2 = IPTS - NPTS2
```

```
C        IF ALPHA.NE.1, COMPUTE OMEGA(U,ALPHA) AND ITS DERIVATIVE W. R.
C        TO ALPHA OUTSIDE MAIN LOOP.
      IF (ALF1) GO TO 10
      OMEGA = TAN(PIBY2*ALPHA)
      PISEC2 = PIBY2 * (ONE + OMEGA*OMEGA)
C
C        LOOP OVER POSITIVE GRID PTS (SGN(U) IGNORED), SAVE PSI AND
C        ITS GRADIENT AT ALL GRID PTS (PSI(U) = RE(PHI(U)) +
C         IM(PHI(U)) - EMPIRICAL COUNTERPARTS.)
   10 DO 20 I = ILOW, NPTS
      IND1 = IND - I
      ISUB = NPTS2 + I
      PTSI = W(ISUB)
      XLOGSU = SIGMA * PTSI
      SUEXP = XLOGSU ** ALPHA
      XLOGSU = ALOG(XLOGSU)
      IF (ALF1) OMEGA = XLOGSU / PIBY2
      COSINE = XMU * PTSI - SUEXP * BSIGN * OMEGA
      SINE = SIN(COSINE)
      COSINE = COS(COSINE)
      XMOD = EXP(-SUEXP)
      SUEXP = SUEXP * XMOD
C
C        SAVE COMPONENTS OF PSI
      ISUB = IPSI + I
      ISUB1 = IECF + I
      W(ISUB) = XMOD * (COSINE + SINE) - W(ISUB1)
      ISUB = IPSI + IND1
      ISUB1 = IECF + IND1
      W(ISUB) = XMOD * (COSINE - SINE) - W(ISUB1)
      IF (LFLAG) GO TO 20
C
C        CALCULATE DERIVATIVES IF REQUIRED
C
C        DERIVATIVES W. R. TO ALPHA
      FAC = XLOGSU * OMEGA
      IF ( .NOT. ALF1) FAC = FAC + PISEC2
      FAC = FAC * BSIGN
      Z = FAC + XLOGSU
      Z1 = FAC - XLOGSU
      ISUB = IDERIV + I
      W(ISUB) = SUEXP * (-Z*COSINE + Z1*SINE)
      ISUB1 = IDERIV + IND1
      W(ISUB1) = SUEXP * (Z1*COSINE + Z*SINE)
C
C        DERIVATIVES W. R. TO BETA
      FAC = SUEXP * OMEGA
      ISUB = ISUB + NPTS
      W(ISUB) = FAC * (SINE - COSINE)
      ISUB1 = ISUB1 + NPTS
      W(ISUB1) = FAC * (SINE + COSINE)
C
C        DERIVATIVES W. R. TO SIGMA
      FAC = BSIGN * OMEGA
      Z = ONE + FAC
      Z1 = ONE - FAC
      FAC = -ALPHA * SUEXP / SIGMA
      ISUB = ISUB + NPTS
      W(ISUB) = FAC * (Z*COSINE + Z1*SINE)
      ISUB1 = ISUB1 + NPTS
```

```
      W(ISUB1) = FAC * (Z1*COSINE - Z*SINE)                             FUNC 096
C                                                                       FUNC 097
C        DERIVATIVES W. R. TO MU                                        FUNC 098
      FAC = PTSI * XMOD                                                 FUNC 099
      ISUB = ISUB + NPTS                                                FUNC 100
      W(ISUB) = FAC * (-SINE + COSINE)                                  FUNC 101
      ISUB1 = ISUB1 + NPTS                                              FUNC 102
      W(ISUB1) = -FAC * (SINE + COSINE)                                 FUNC 103
   20 CONTINUE                                                          FUNC 104
C                                                                       FUNC 105
C        NOW COMPUTE OBJECTIVE FUNCTION , OPTIONALLY GRADIENT.          FUNC 106
      FC = ZERO                                                         FUNC 107
      IF (LFLAG) GO TO 40                                               FUNC 108
      ILOW = IDERIV - NPTS                                              FUNC 109
      DO 30 I = 1, N                                                    FUNC 110
   30 GC(I) = ZERO                                                      FUNC 111
   40 IF (IW(3) .EQ. 0) GO TO 70                                        FUNC 112
C                                                                       FUNC 113
C        SUM-OF-SQUARES ESTIMATION                                      FUNC 114
      DO 60 I = 1, NPTS                                                 FUNC 115
C        FN. EVALUATION                                                 FUNC 116
      ISUB = IPSI + I                                                   FUNC 117
      Z = W(ISUB)                                                       FUNC 118
      ISUB = IWT + I                                                    FUNC 119
      Z1 = Z * W(ISUB)                                                  FUNC 120
      FC = FC + Z * Z1                                                  FUNC 121
      IF (LFLAG) GO TO 60                                               FUNC 122
C        GRADIENT EVALUATION IF REQUESTED.                              FUNC 123
      ISUB = ILOW + I                                                   FUNC 124
      Z1 = Z1 + Z1                                                      FUNC 125
      DO 50 J = 1, N                                                    FUNC 126
      ISUB = ISUB + NPTS                                                FUNC 127
      GC(J) = GC(J) + W(ISUB) * Z1                                      FUNC 128
   50 CONTINUE                                                          FUNC 129
   60 CONTINUE                                                          FUNC 130
      RETURN                                                            FUNC 131
C                                                                       FUNC 132
C        MATRIX ESTIMATION - FIRST MULTIPLY PSI VALUES BY WEIGHTS.      FUNC 133
   70 DO 80 I = 1, NPTS                                                 FUNC 134
      ISUB = IPSI + I                                                   FUNC 135
      ISUB1 = IWT + I                                                   FUNC 136
      W(ISUB) = W(ISUB) * W(ISUB1)                                      FUNC 137
   80 CONTINUE                                                          FUNC 138
C        POSITION OF A INVERSE IS REQUIRED.                             FUNC 139
      IND1 = IPSI + IND * NPTS                                          FUNC 140
      DO 110 I = 1, NPTS                                                FUNC 141
      Z = ZERO                                                          FUNC 142
      IND1 = IND1 + NPTS                                                FUNC 143
C        SUM OVER J OF PSI(J) * AINV(I,J)                               FUNC 144
      DO 90 J = 1, NPTS                                                 FUNC 145
      ISUB = IPSI + J                                                   FUNC 146
      ISUB1 = IND1 + J                                                  FUNC 147
      Z = Z + W(ISUB) * W(ISUB1)                                        FUNC 148
   90 CONTINUE                                                          FUNC 149
C        MULTIPLY BY PSI(I) AND ADD TO FN. VALUE                        FUNC 150
      ISUB = IPSI + I                                                   FUNC 151
      FC = FC + Z * W(ISUB)                                             FUNC 152
      IF (LFLAG) GO TO 110                                              FUNC 153
C                                                                       FUNC 154
C        GRADIENT IF REQUESTED - FOR JTH COMPONENT OF GRADIENT ADD      FUNC 155
```

```
C        2*(JTH DERIVATIVE AT GRIDPOINT I)*WT(I)*(RESULT OF DO 90 LOOP)
      ISUB = IWT + 1
      Z = (Z + Z) * W(ISUB)
      ISUB = ILOW + 1
      DO 100 J = 1, N
      ISUB = ISUB + NPTS
      GC(J) = GC(J) + Z * W(ISUB)
  100 CONTINUE
  110 CONTINUE
C
      RETURN
      END
C**********************************************************************
C        COMPUTES EMPIRICAL CH. F. VALUES, ADJUSTING FOR LOCATION,
C        SCALE, AND K-SUM INDEX.  FOR MATRIX ESTIMATION, THE UPPER
C        TRIANGLE OF A MATRIX IS CALCULATED AND INVERTED.
C        CALLED BY - STABLE
C        CALLS - CHARFN
C        N.A.G. SUBROUTINE CALLED -
C               F01ABF (ACCURATE INVERSION OF POSITIVE DEFINITE
C               SYMMETRIC MATRIX).
C**********************************************************************
      SUBROUTINE SETECF(X, N, PAR, NPAR, MODE, TAU, SIGMA, XMU, KSUM,
     *IA, NPTS2, NPTS, PTS, ECF, A, AINV, WORK, IFAULT)
C        ARGUMENTS
      INTEGER N, NPAR, MODE, KSUM, IA, NPTS2, NPTS, IFAULT
      REAL X(N), PAR(NPAR), TAU, SIGMA, XMU, PTS(NPTS2), ECF(NPTS),
     *A(IA,NPTS), AINV(NPTS,NPTS), WORK(NPTS)
C        LOCAL SCALARS
      INTEGER IND, IND1, IND2, IND3
      REAL PTSI, PTSJ, RE, XIM, RE1, XIM1, RPI, RMI, RPI1, RMI1, ZERO,
     *HALF
      DATA ZERO, HALF /0.0, 0.5/
C
C        CALCULATION OF E.CH.F. VALUES
      PTSJ = FLOAT(N)
      XIM1 = FLOAT(KSUM)
      RPI = XIM1 * HALF
      IND = NPTS2 + 1
      IND1 = NPTS + 1
      DO 20 I = IND, NPTS
      IND2 = I - NPTS2
      PTSI = PTS(IND2) / SIGMA
      IND2 = IND1 - I
      RE = ZERO
      XIM = ZERO
C
      DO 10 J = 1, N
      RPI1 = X(J) * PTSI
      RE = RE + COS(RPI1)
      XIM = XIM + SIN(RPI1)
   10 CONTINUE
C
      RE = RE / PTSJ
      XIM = XIM / PTSJ
      RMI = (RE*RE + XIM*XIM) ** RPI
      RPI1 = XIM1 * ATAN2(XIM,RE) - XMU * PTSI
      RE = RMI * COS(RPI1)
      XIM = RMI * SIN(RPI1)
      ECF(I) = RE + XIM
```

```
      ECF(IND2) = RE - XIM                                              SETE 049
   20 CONTINUE                                                          SETE 050
C                                                                       SETE 051
C         SET UPPER TRIANGLE OF A IF REQUESTED.  A GENERATED FROM       SETE 052
C         POSITIVE GRIDPOINTS ONLY - ANTIDIAGONAL COMPUTED TWICE.       SETE 053
      IF (MODE .NE. 0) RETURN                                           SETE 054
C         FIRST FILL WORK WITH (RE + IM) (PHI) TO SAVE EVALS.           SETE 055
      DO 30 I = IND, NPTS                                               SETE 056
      IND2 = I - NPTS2                                                  SETE 057
      CALL CHARFN(PTS(IND2), PAR, NPAR, RE, XIM)                        SETE 058
      IND2 = IND1 - I                                                   SETE 059
      WORK(I) = RE + XIM                                                SETE 060
      WORK(IND2) = RE - XIM                                             SETE 061
   30 CONTINUE                                                          SETE 062
C         COMPUTATION OF A.                                             SETE 063
      DO 40 I = IND, NPTS                                               SETE 064
      IND2 = I - NPTS2                                                  SETE 065
      PTSI = PTS(IND2)                                                  SETE 066
      IND2 = IND1 - I                                                   SETE 067
      RPI = WORK(I)                                                     SETE 068
      RMI = WORK(IND2)                                                  SETE 069
      DO 40 J = IND, I                                                  SETE 070
      IND3 = J - NPTS2                                                  SETE 071
      PTSJ = PTS(IND3)                                                  SETE 072
      IND3 = IND1 - J                                                   SETE 073
      RPI1 = WORK(J)                                                    SETE 074
      RMI1 = WORK(IND3)                                                 SETE 075
      CALL CHARFN(PTSI + PTSJ, PAR, NPAR, RE, XIM)                      SETE 076
      CALL CHARFN(PTSI - PTSJ, PAR, NPAR, RE1, XIM1)                    SETE 077
      A(J,I) = RE1 + XIM - RPI * RPI1                                   SETE 078
      A(IND2,J) = RE - XIM1 - RMI * RPI1                                SETE 079
      A(IND3,I) = RE + XIM1 - RPI * RMI1                                SETE 080
      A(IND2,IND3) = RE1 - XIM - RMI * RMI1                             SETE 081
   40 CONTINUE                                                          SETE 082
C                                                                       SETE 083
C         DIAGONAL OF A ADDITIVELY INFLATED BY TAU TIMES AVERAGE OF     SETE 084
C         DIAGONAL ELEMENTS.                                            SETE 085
      IF (TAU .EQ. ZERO) GO TO 70                                       SETE 086
      PTSI = ZERO                                                       SETE 087
      DO 50 I = 1, NPTS                                                 SETE 088
   50 PTSI = PTSI + A(I,I)                                              SETE 089
      PTSI = TAU * PTSI / FLOAT(NPTS)                                   SETE 090
      DO 60 I = 1, NPTS                                                 SETE 091
   60 A(I,I) = A(I,I) + PTSI                                            SETE 092
C                                                                       SETE 093
C         INVERT A USING N.A.G. ROUTINE F01ABF                          SETE 094
   70 IFAULT = 1                                                        SETE 095
      CALL F01ABF(A, IA, NPTS, AINV, NPTS, WORK, IFAULT)                SETE 096
C         EXIT IF A FOUND NON POSITIVE DEFINITE OR ILL-CONDITIONED.     SETE 097
      IF (IFAULT .GT. 0) RETURN                                         SETE 098
C         ARRANGE A INVERSE SO IT IS COMPLETELY FILLED.                 SETE 099
      DO 80 I = 1, NPTS                                                 SETE 100
      DO 80 J = 1, I                                                    SETE 101
      AINV(J,I) = AINV(I,J)                                             SETE 102
   80 CONTINUE                                                          SETE 103
C                                                                       SETE 104
      RETURN                                                            SETE 105
      END                                                               SETE 106
C***********************************************************************MONI 001
C       MONITORING OF E04KBF.                                           MONI 002
```

```
C       CALLED BY - E04KBF
C***********************************************************************
      SUBROUTINE MONIT(N, XC, FC, GC, ISTATE, GPJNRM, COND, POSDEF,
     *NITER, NF, IW, LIW, W, LW)
C         ARGUMENTS
      LOGICAL POSDEF
      INTEGER N, ISTATE(N), NITER, NF, LIW, IW(LIW), LW
      REAL XC(N), FC, GC(N), GPJNRM, COND, W(LW)
C         LOCAL SCALAR
      INTEGER IUNIT
C
C         STORE HESSIAN CONDITION NUMBER AND PROJECTED GRADIENT NORM IN
C         WORK VECTOR
      IUNIT = 9 * N + 1
      W(IUNIT) = COND
      W(IUNIT + 1) = GPJNRM
      IUNIT = IW(4)
      IF (IUNIT .LE. 0) RETURN
C
C         WRITE DETAILS OF OPTIMIZATION PROCESS IF IUNIT .GT. 0
      WRITE (IUNIT,10) NITER, NF, FC, GPJNRM
      WRITE (IUNIT,20) (XC(I),I=1,N)
      WRITE (IUNIT,30) (GC(I),I=1,N)
      WRITE (IUNIT,40) COND
C
   10 FORMAT (5HOITNS, 5X, 8HFN EVALS, 8X, 8HFN VALUE, 5X,
     *21HNORM OF PROJ GRADIENT/1H , I4, 8X, I5, 5X, E11.4, 15X, E11.4)
   20 FORMAT (10HOSOLUTIONS, 4E13.5)
   30 FORMAT (10H PROJ GRAD, 4E13.5)
   40 FORMAT (50H ESTIMATED CONDITION NUMBER OF PROJECTED HESSIAN =,
     *E12.2)
      RETURN
      END
C***********************************************************************
C         CALCULATES EMPIRICAL (VCV1) AND ASYMPTOTIC (VCV2)
C         APPROXIMATIONS TO ASYMPTOTIC COVARIANCE MATRICES.  IFAIL1 AND
C         IFAIL2 ARE FAILURE INDICATORS FOR MATRIX INVERSION REQUIRED
C         FOR VCV1, VCV2 RESPECTIVELY.
C         CALLED BY - STABLE
C         CALLS - HESDIF, FUNCT, CHARFN, SETVCV, VMATRX, DAPROD, HVPROD
C         N.A.G. SUBROUTINES CALLED -
C                FO1CAF (SETS A MATRIX TO ZERO),
C                FO1CMF (SET ONE MATRIX TO ANOTHER).
C***********************************************************************
      SUBROUTINE VARIAB(ICOV, X, N, PAR, NPAR, MODE, SIGMA, XMU, ISUB,
     *NVAR, PTS, NPTS2, WT, ECF, NPTS, DERIV, WORK, HOLD, A, IA, AINV,
     *VCV1, VCV2, H, NVAR1, V, IW, LIW, W, LW, IFAIL1, IFAIL2)
C         ARGUMENTS
      INTEGER ICOV, N, NPAR, MODE, NVAR, ISUB(NVAR), NPTS2, NPTS, IA,
     *NVAR1, LIW, IW(LIW), LW, IFAIL1, IFAIL2
      REAL X(N), PAR(NPAR), SIGMA, XMU, PTS(NPTS2), WT(NPTS), ECF(NPTS),
     *DERIV(NPTS,NPAR), WORK(NPTS), HOLD(NVAR), A(IA,NPTS),
     *AINV(NPTS,NPTS), VCV1(NPAR,NPAR), VCV2(NPAR,NPAR), H(NVAR1,NVAR),
     *V(NVAR,NVAR), W(LW)
C         LOCAL SCALARS
      LOGICAL FLAG
      INTEGER IND, IND1, IND2, IND3
      REAL PTSK, PTSL, RPI, RMI, RPI1, RMI1, COVKL, COVML, COVKM, COVMM,
     *D1, D2, D3, D4, ZERO
      DATA ZERO /0.0/
```

```
C                                                                       VARI 028
      IND = NPTS2 + 1                                                   VARI 029
      IND1 = NPTS + 1                                                   VARI 030
C                                                                       VARI 031
C         IF REQUESTED, FIRST CALCULATE HESSIAN FOR EMPIRICAL VERSION   VARI 032
C         AND STORE IN VCV1.  (HESDIF, FUNCT REQUIRE ECF, WORK,         VARI 033
C         POSSIBLY AINV, WHICH ARE USED AS WORK AREAS BELOW.)           VARI 034
C         H AND V ARE PASSED AS WORKSPACE.                              VARI 035
      IF (ICOV .EQ. 0) CALL HESDIF(PAR, NPAR, ISUB, VCV1, H, V, NVAR,   VARI 036
     *IW, LIW, W, LW)                                                   VARI 037
C                                                                       VARI 038
C         CALL FUNCT TO SET DERIV, AS GRID SEARCH PERFORMED BY E04KBF   VARI 039
C         MAY HAVE SET IT TO STRANGE VALUES.  VCV2 USED AS WORKSPACE.   VARI 040
      CALL FUNCT(2, NPAR, PAR, D1, VCV2(1,1), IW, LIW, W, LW)           VARI 041
C                                                                       VARI 042
C         SHIFT DERIV VALUES SO THEY CAN BE ADDRESSED WITHOUT THE       VARI 043
C         ISUB VECTOR.  NOTE ISUB ELEMENTS ARE IN ASCENDING ORDER.      VARI 044
      DO 20 I = 1, NVAR                                                 VARI 045
      IND2 = ISUB(I)                                                    VARI 046
      IF (IND2 .EQ. I) GO TO 20                                         VARI 047
      DO 10 J = 1, NPTS                                                 VARI 048
   10 DERIV(J,I) = DERIV(J,IND2)                                        VARI 049
   20 CONTINUE                                                          VARI 050
C                                                                       VARI 051
C         FILL ECF VECTOR WITH CH. F. VALUES, NOW THAT IT IS NO         VARI 052
C         LONGER NEEDED FOR FUNCTION EVALUATION.                        VARI 053
      DO 30 I = IND, NPTS                                               VARI 054
      IND2 = I - NPTS2                                                  VARI 055
      CALL CHARFN(PTS(IND2), PAR, NPAR, D1, D2)                         VARI 056
      IND2 = IND1 - I                                                   VARI 057
      ECF(I) = D1 + D2                                                  VARI 058
      ECF(IND2) = D1 - D2                                               VARI 059
   30 CONTINUE                                                          VARI 060
      IF (MODE .EQ. 0) GO TO 100                                        VARI 061
C                                                                       VARI 062
C         SUM OF SQUARES ESTIMATION SECTION.                            VARI 063
C                                                                       VARI 064
C         FIRST DO ASYMPTOTIC VERSION.                                  VARI 065
C         CALCULATE UPPER TRIANGLE OF HESSIAN.                          VARI 066
      DO 50 I = 1, NVAR                                                 VARI 067
      DO 50 J = I, NVAR                                                 VARI 068
      D1 = ZERO                                                         VARI 069
      DO 40 K = 1, NPTS                                                 VARI 070
   40 D1 = D1 + DERIV(K,I) * DERIV(K,J) * WT(K)                         VARI 071
      H(I,J) = D1                                                       VARI 072
   50 CONTINUE                                                          VARI 073
C                                                                       VARI 074
C         PREMULTIPLY DERIV BY WEIGHTS TO SAVE MULTIPLICATIONS.         VARI 075
      DO 60 I = 1, NPTS                                                 VARI 076
      D1 = WT(I)                                                        VARI 077
      DO 60 J = 1, NVAR                                                 VARI 078
      DERIV(I,J) = D1 * DERIV(I,J)                                      VARI 079
   60 CONTINUE                                                          VARI 080
C                                                                       VARI 081
C         COMPUTE UPPER TRIANGLE OF V.  BASICALLY EQUIVALENT TO         VARI 082
C         CALCULATING THE A MATRIX FOR MATRIX ESTIMATION, BUT           VARI 083
C         COMPLICATIONS ARISE IN COMPUTING BILINEAR FORMS.              VARI 084
      CALL F01CAF(V, NVAR, NVAR, IFAIL2)                                VARI 085
      DO 90 K = IND, NPTS                                               VARI 086
      IND2 = K - NPTS2                                                  VARI 087
```

```
      PTSK = PTS(IND2)
      IND2 = IND1 - K
      RPI = ECF(K)
      RMI = ECF(IND2)
      DO 90 L = IND, K
      IND3 = L - NPTS2
      PTSL = PTS(IND3)
      IND3 = IND1 - L
      RPI1 = ECF(L)
      RMI1 = ECF(IND3)
      CALL CHARFN(PTSK + PTSL, PAR, NPAR, D1, D2)
      CALL CHARFN(PTSK - PTSL, PAR, NPAR, D3, D4)
      COVKL = D3 + D2 - RPI * RPI1
      COVML = D1 - D4 - RMI * RPI1
      COVKM = D1 + D4 - RPI * RMI1
      COVMM = D3 - D2 - RMI * RMI1
      FLAG = K .EQ. L
C         LOOP TO ADD CONTRIBUTIONS TO V (BILINEAR FORMS).
      DO 80 I = 1, NVAR
      D1 = DERIV(K,I)
      D2 = DERIV(IND2,I)
      DO 80 J = 1, NVAR
      D3 = DERIV(L,J)
      D4 = DERIV(IND3,J)
      PTSL = (COVKL*D1 + COVML*D2) * D3 + (COVKM*D1 + COVMM*D2) * D4
      IF (FLAG) GO TO 70
C
C         WHEN K .NE. L , MUST ADD SYMMETRIC CONTRIBUTION.
      RPI1 = DERIV(K,J)
      RMI1 = DERIV(IND2,J)
      D3 = DERIV(L,I)
      D4 = DERIV(IND3,I)
      PTSL = PTSL + (COVKL*RPI1 + COVML*RMI1) * D3 + (COVKM*RPI1 +
     *COVMM*RMI1) * D4
   70 V(I,J) = V(I,J) + PTSL
   80 CONTINUE
   90 CONTINUE
      GO TO 140
C
C         MATRIX ESTIMATION SECTION.
C
C         FILL A MATRIX COMPLETELY (UPPER TRIANGLE ONLY ON ENTRY),
C         MULTIPLY ELEMENTS OF A BY CORRESPONDING WEIGHTS, MULTIPLY
C         DERIV VALUES BY WEIGHTS TO SAVE MULTIPLICATIONS LATER.
  100 DO 130 I = 1, NPTS
      D1 = WT(I)
      DO 110 J = 1, NPTS
      A(I,J) = A(I,J) * D1 * WT(J)
      A(J,I) = A(I,J)
  110 CONTINUE
      DO 120 J = 1, NVAR
  120 DERIV(I,J) = DERIV(I,J) * D1
  130 CONTINUE
C
C         ASYMPTOTIC VERSION.
C         MULTIPLY AINV * DERIV, OVERWRITING CORNER OF AINV AND USING
C         HOLD AS WORKSPACE.
      CALL DAPROD(AINV, NPTS, NPTS, DERIV, HOLD, NVAR)
C
C         MULTIPLY A * (AINV CORNER), OVERWRITING CORNER OF A AND
```

```
      DO 50 I = 1, N
      D1 = (X(I) - XMU) / SIGMA
C
C         WORK HOLDS VECTOR OF SINE/COSINE TERMS.
      DO 10 J = IND, NPTS
      IND2 = J - NPTS2
      D2 = PTS(IND2) * D1
      D3 = COS(D2)
      D2 = SIN(D2)
      IND2 = IND1 - J
      WORK(J) = ECF(J) - D3 - D2
      WORK(IND2) = ECF(IND2) - D3 + D2
      IF (FLAG) GO TO 10
C
C         IF MATRIX ESTIMATION, MULTIPLY ELEMENTS OF WORK BY WEIGHTS.
      WORK(J) = WORK(J) * WT(J)
      WORK(IND2) = WORK(IND2) * WT(IND2)
   10 CONTINUE
C
C         MULTIPLY DERIV * WORK, WRITING RESULT TO HOLD.  MULTIPLICATION
C         DONE DIRECTLY TO AVOID OVERHEAD OF SUBROUTINE CALL.
      DO 30 J = 1, NVAR
      D1 = ZERO
      DO 20 K = 1, NPTS
   20 D1 = D1 + DERIV(K,J) * WORK(K)
      HOLD(J) = D1
   30 CONTINUE
C
C         ADD HOLD * (HOLD TRANSPOSE) TO V.
      DO 40 J = 1, NVAR
      D1 = HOLD(J)
      DO 40 K = J, NVAR
      V(J,K) = V(J,K) + D1 * HOLD(K)
   40 CONTINUE
   50 CONTINUE
C
C         FINAL CONSTANT FACTOR FOR V.
      D1 = FOUR / FLOAT(N)
      DO 60 I = 1, NVAR
      DO 60 J = 1, NVAR
   60 V(I,J) = V(I,J) * D1
C
      RETURN
      END
C***********************************************************************
C         MULTIPLIES THE (NPTS BY NPTS) CORNER OF THE (IFAC1 BY NPTS)
C         MATRIX FAC1 BY THE (NPTS BY NVAR) MATRIX FAC2, OVERWRITING
C         THE UPPER LEFT (NPTS BY NVAR) ELEMENTS OF FAC1.
C         UNFORTUNATELY, N.A.G. ROUTINE F01CKF DOES NOT ALLOW THIS
C         KIND OF OVERWRITING.
C         CALLED BY - VARIAB
C***********************************************************************
      SUBROUTINE DAPROD(FAC1, IFAC1, NPTS, FAC2, WORK, NVAR)
C         ARGUMENTS
      INTEGER IFAC1, NPTS, NVAR
      REAL FAC1(IFAC1,NPTS), FAC2(NPTS,NVAR), WORK(NVAR)
C         LOCAL SCALARS
      REAL TEMP, ZERO
      DATA ZERO /0.0/
C
```

```
C        USING HOLD AS WORKSPACE.
      CALL DAPROD(A, IA, NPTS, AINV, HOLD, NVAR)
C
C        MULTIPLY (A CORNER TRANSPOSE) * (AINV CORNER), GIVING
C        UPPER TRIANGLE OF V.
      CALL HVPROD(A, IA, NVAR, AINV, NPTS, V, NVAR)
C
C        MULTIPLY (AINV CORNER TRANSPOSE) * DERIV, GIVING UPPER
C        TRIANGLE OF HESSIAN.
      CALL HVPROD(AINV, NPTS, NVAR, DERIV, NPTS, H, NVAR1)
C
C        CODE COMMON TO MATRIX AND SUM OF SQUARES ESTIMATION.
C        COMPUTE ASYMPTOTIC COVARIANCE MATRIX.
  140 CALL SETVCV(ISUB, NVAR, H, NVAR1, V, HOLD, VCV2, NPAR, SIGMA,
     *IFAIL2)
      IF (ICOV .GT. 0) RETURN
C
C        EMPIRICAL VERSION IF REQUESTED.
C        COMPUTE EMPIRICAL V MATRIX, USING WORK AND HOLD AS WORK AREAS.
      IF (MODE .NE. 0) CALL VMATRX(X, N, MODE, XMU, SIGMA, PTS, NPTS2,
     *WT, ECF, WORK, NPTS, DERIV, V, HOLD, NVAR)
      IF (MODE .EQ. 0) CALL VMATRX(X, N, MODE, XMU, SIGMA, PTS, NPTS2,
     *WT, ECF, WORK, NPTS, AINV, V, HOLD, NVAR)
C
C        RESTORE HESSIAN FROM VCV1, COMPUTE COVARIANCE MATRIX.
      CALL F01CMF(VCV1, NVAR, H, NVAR1, NVAR, NVAR)
      CALL SETVCV(ISUB, NVAR, H, NVAR1, V, HOLD, VCV1, NPAR, SIGMA,
     *IFAIL1)
C
      RETURN
      END
C***********************************************************************
C        CALCULATES EMPIRICAL V MATRIX.  IMPLICIT IS COMPUTATION OF
C        EMPIRICAL COVARIANCE KERNEL, BUT IT IS FASTER TO COMPUTE A
C        MATRIX PRODUCT FOR EACH SAMPLE POINT THAN TO CUMULATE THE
C        WHOLE KERNEL.  FOR MATRIX ESTIMATION, THE DERIV ARRAY
C        IS ACTUALLY THE UPPER LEFT CORNER OF AINV.
C        CALLED BY - VARIAB
C        N.A.G. SUBROUTINE CALLED -
C               F01CAF (SET A MATRIX TO 0).
C***********************************************************************
      SUBROUTINE VMATRX(X, N, MODE, XMU, SIGMA, PTS, NPTS2, WT, ECF,
     *WORK, NPTS, DERIV, V, HOLD, NVAR)
C        ARGUMENTS
      INTEGER N, MODE, NPTS2, NPTS, NVAR
      REAL X(N), XMU, SIGMA, PTS(NPTS2), WT(NPTS), ECF(NPTS),
     *WORK(NPTS), DERIV(NPTS,NVAR), V(NVAR,NVAR), HOLD(NVAR)
C        LOCAL SCALARS
      LOGICAL FLAG
      INTEGER IND, IND1, IND2
      REAL D1, D2, D3, ZERO, FOUR
      DATA ZERO, FOUR /0.0, 4.0/
C
C        SET V TO 0.
      FLAG = MODE .NE. 0
      IND = NPTS2 + 1
      IND1 = NPTS + 1
      CALL F01CAF(V, NVAR, NVAR, IND2)
C
C        MAIN LOOP OVER SAMPLE.
```

```
      DO 40 I = 1, NPTS
      DO 20 J = 1, NVAR
      TEMP = ZERO
      DO 10 K = 1, NPTS
   10 TEMP = TEMP + FAC1(I,K) * FAC2(K,J)
      WORK(J) = TEMP
   20 CONTINUE
      DO 30 J = 1, NVAR
   30 FAC1(I,J) = WORK(J)
   40 CONTINUE
C
      RETURN
      END
C***********************************************************************
C         MULTIPLIES THE TRANSPOSED (NPTS BY NVAR) CORNER OF THE
C         (IFAC1 BY NVAR) MATRIX FAC1 BY THE (NPTS BY NVAR) MATRIX
C         FAC2, GIVING THE UPPER TRIANGLE OF EITHER V OR H.
C         CALLED BY - VARIAB
C***********************************************************************
      SUBROUTINE HVPROD(FAC1, IFAC1, NVAR, FAC2, NPTS, VH, IVH)
C         ARGUMENTS
      INTEGER IFAC1, NVAR, NPTS, IVH
      REAL FAC1(IFAC1,NVAR), FAC2(NPTS,NVAR), VH(IVH,NVAR)
C         LOCAL SCALARS
      REAL TEMP, ZERO
      DATA ZERO /0.0/
C
      DO 20 I = 1, NVAR
      DO 20 J = I, NVAR
      TEMP = ZERO
      DO 10 K = 1, NPTS
   10 TEMP = TEMP + FAC1(K,I) * FAC2(K,J)
      VH(I,J) = TEMP
   20 CONTINUE
C
      RETURN
      END
C***********************************************************************
C         COMMON OPERATIONS IN COMPUTATION OF COVARIANCE MATRICES.
C         INVERTS HESSIAN MATRIX H, CALCULATES (H INVERSE) * V *
C         (H INVERSE), OVERWRITING V.
C         CALLED BY - VARIAB
C         N.A.G. SUBROUTINES CALLED -
C                 F01ABF (ACCURATE INVERSION OF POSITIVE DEFINITE
C                        SYMMETRIC MATRIX),
C                 F01CKF (MATRIX MULTIPLICATION WITH OVERWRITING),
C                 F01CMF (SET ONE MATRIX EQUAL TO ANOTHER),
C                 F01CAF (SET A MATRIX TO ZERO).
C***********************************************************************
      SUBROUTINE SETVCV(ISUB, NVAR, H, NVAR1, V, WORK, VCV, NPAR, SIGMA,
     *IFAULT)
C         ARGUMENTS
      INTEGER NVAR, ISUB(NVAR), NVAR1, NPAR, IFAULT
      REAL H(NVAR1,NVAR), V(NVAR,NVAR), WORK(NVAR), VCV(NPAR,NPAR),
     *SIGMA
C         LOCAL SCALARS
      INTEGER IND, IND1
      REAL TEMP, ZERO
      DATA ZERO /0.0/
C
```

```
C         INVERT H, USING VCV AS WORKSPACE.                              SETV 024
      IFAULT = 1                                                         SETV 025
      CALL F01ABF(H, NVAR1, NVAR, VCV, NPAR, WORK, IFAULT)               SETV 026
      IF (IFAULT .EQ. 0) GO TO 10                                        SETV 027
C         ON FAILURE OF INVERSION, SET VCV TO 0 AND RETURN.              SETV 028
      CALL F01CAF(VCV, NPAR, NPAR, IND)                                  SETV 029
      RETURN                                                             SETV 030
C         FILL OUT VCV AND V (CURRENTLY ONLY HALF FULL).                 SETV 031
   10 DO 20 I = 1, NVAR                                                  SETV 032
      DO 20 J = 1, I                                                     SETV 033
      V(I,J) = V(J,I)                                                    SETV 034
      VCV(J,I) = VCV(I,J)                                                SETV 035
   20 CONTINUE                                                           SETV 036
C                                                                        SETV 037
C         SET H TO ITS INVERSE IN SUCH A WAY THAT MULTIPLICATION VIA     SETV 038
C         F01CKF WILL BE CORRECT.  (NOTE DIMENSIONS IN F01CMF CALL)      SETV 039
      CALL F01CMF(VCV, NPAR, H, NVAR, NVAR, NVAR)                        SETV 040
C                                                                        SETV 041
C         MULTIPLY H * V, OVERWRITING V.                                 SETV 042
      CALL F01CKF(V, H, V, NVAR, NVAR, NVAR, WORK, NVAR, 3, IFAULT)      SETV 043
C                                                                        SETV 044
C         MULTIPLY V * H, OVERWRITING V.                                 SETV 045
      CALL F01CKF(V, V, H, NVAR, NVAR, NVAR, WORK, NVAR, 2, IFAULT)      SETV 046
C                                                                        SETV 047
C         V NOW CONTAINS COVARIANCE MATRIX FOR FREE VARIABLES.  ARRANGE  SETV 048
C         ITS CONTENTS IN VCV, ADJUST SIGMA AND MU ENTRIES FOR SCALE.    SETV 049
      CALL F01CAF(VCV, NPAR, NPAR, IFAULT)                               SETV 050
      DO 30 I = 1, NVAR                                                  SETV 051
      DO 30 J = 1, I                                                     SETV 052
      IND = MIN0(ISUB(I),ISUB(J))                                        SETV 053
      IND1 = MAX0(ISUB(I),ISUB(J))                                       SETV 054
      VCV(IND,IND1) = V(I,J)                                             SETV 055
   30 CONTINUE                                                           SETV 056
      DO 40 J = 3, NPAR                                                  SETV 057
      DO 40 I = 1, J                                                     SETV 058
      TEMP = SIGMA * VCV(I,J)                                            SETV 059
      IF (I .GE. 3) TEMP = SIGMA * TEMP                                  SETV 060
      VCV(I,J) = TEMP                                                    SETV 061
   40 CONTINUE                                                           SETV 062
C                                                                        SETV 063
C         FILL LOWER TRIANGLE OF VCV WITH CORRELATIONS.                  SETV 064
      DO 50 I = 2, NPAR                                                  SETV 065
      TEMP = VCV(I,I)                                                    SETV 066
      IND = I - 1                                                        SETV 067
      DO 50 J = 1, IND                                                   SETV 068
      IF (AMIN1(TEMP,VCV(J,J)) .LE. ZERO) GO TO 50                       SETV 069
      VCV(I,J) = VCV(J,I) / SQRT(TEMP*VCV(J,J))                          SETV 070
   50 CONTINUE                                                           SETV 071
C                                                                        SETV 072
      RETURN                                                             SETV 073
      END                                                                SETV 074
C***********************************************************************HESD 001
C         COMPUTES APPROXIMATE UPPER TRIANGLE OF HESSIAN BY DIFFERENCES. HESD 002
C         NOTE THAT WITH IFLAG = 0, FUNCT WILL NOT SET A GRADIENT.       HESD 003
C         CALLED BY - VARIAB                                             HESD 004
C         CALLS - FUNCT                                                  HESD 005
C***********************************************************************HESD 006
      SUBROUTINE HESDIF(PAR, NPAR, ISUB, H, SAVE1, SAVE2, NVAR, IW, LIW,HESD 007
     *W, LW)                                                             HESD 008
C         ARGUMENTS                                                      HESD 009
```

```
      INTEGER NPAR, NVAR, ISUB(NVAR), LIW, IW(LIW), LW                    HESD 010
      REAL PAR(NPAR), H(NVAR,NVAR), SAVE1(NVAR,NVAR), SAVE2(NVAR,NVAR),   HESD 011
     *W(LW)                                                               HESD 012
C          LOCAL SCALARS                                                  HESD 013
      INTEGER ITER, IND, IND1, IND2, IND3                                 HESD 014
      REAL PARI, PARJ, TEMP, TEMP1, STEP, DENOM, ZERO, TOL, TENM4,        HESD 015
     *STEP1, ONE, TWO, SQRT10, FOUR                                       HESD 016
      DATA ZERO, TOL, TENM4, STEP1, ONE, TWO, SQRT10, FOUR /0.0, 1.0E-6,HESD 017
     *1.0E-4, 3.162277660E-3, 1.0, 2.0, 3.162277660, 4.0/                 HESD 018
C                                                                         HESD 019
C          START WITH STEPLENGTH 1.0E-3, REPEATEDLY DIVIDE BY             HESD 020
C          SQRT10. ITERATION STOPS WHEN DIFFERENCE BETWEEN SUCCESSIVE     HESD 021
C          HESSIANS LESS THAN 1.0E-6. IF NO SUCCESS IN 5 ITERATIONS,      HESD 022
C          USE RESULT WITH STEPLENGTH 1.0E-4.                             HESD 023
C          DIFFERENCE IS MAX. DIFFERENCE BETWEEN SUCCESSIVE ELEMENTS-     HESD 024
C            ABSOLUTE DIFFERENCE IF /LATEST ELEMENT/ .LT. 1,              HESD 025
C            RELATIVE DIFFERENCE IF /LATEST ELEMENT/ .GE. 1.              HESD 026
      ITER = 0                                                            HESD 027
      STEP = STEP1                                                        HESD 028
      DO 10 I = 1, NVAR                                                   HESD 029
      DO 10 J = 1, NVAR                                                   HESD 030
   10 SAVE1(I,J) = ZERO                                                   HESD 031
C                                                                         HESD 032
C          STARTING POINT FOR ITERATION.                                  HESD 033
   20 ITER = ITER + 1                                                     HESD 034
      STEP = STEP / SQRT10                                                HESD 035
C                                                                         HESD 036
C          DIAGONAL ELEMENTS. THREE POINT DIFFERENCING.                   HESD 037
      DENOM = STEP * STEP                                                 HESD 038
      DO 30 I = 1, NVAR                                                   HESD 039
      IND = ISUB(I)                                                       HESD 040
      PARI = PAR(IND)                                                     HESD 041
      PAR(IND) = PARI + STEP                                              HESD 042
      CALL FUNCT(0, NPAR, PAR, TEMP, PAR, IW, LIW, W, LW)                 HESD 043
      PAR(IND) = PARI - STEP                                              HESD 044
      CALL FUNCT(0, NPAR, PAR, TEMP1, PAR, IW, LIW, W, LW)                HESD 045
      TEMP = TEMP + TEMP1                                                 HESD 046
      PAR(IND) = PARI                                                     HESD 047
      CALL FUNCT(0, NPAR, PAR, TEMP1, PAR, IW, LIW, W, LW)                HESD 048
      H(I,I) = (TEMP - TWO*TEMP1) / DENOM                                 HESD 049
   30 CONTINUE                                                            HESD 050
      IF (NVAR .EQ. 1) GO TO 50                                           HESD 051
C                                                                         HESD 052
C          OFF-DIAGONAL ELEMENTS IF REQUIRED. FOUR POINT DIFFERENCING.    HESD 053
      DENOM = FOUR * DENOM                                                HESD 054
      IND = NVAR - 1                                                      HESD 055
      DO 40 I = 1, IND                                                    HESD 056
      IND1 = I + 1                                                        HESD 057
      IND2 = ISUB(I)                                                      HESD 058
      PARI = PAR(IND2)                                                    HESD 059
      DO 40 J = IND1, NVAR                                                HESD 060
      IND3 = ISUB(J)                                                      HESD 061
      PARJ = PAR(IND3)                                                    HESD 062
      PAR(IND2) = PARI + STEP                                             HESD 063
      PAR(IND3) = PARJ + STEP                                             HESD 064
      CALL FUNCT(0, NPAR, PAR, TEMP, PAR, IW, LIW, W, LW)                 HESD 065
      PAR(IND3) = PARJ - STEP                                             HESD 066
      CALL FUNCT(0, NPAR, PAR, TEMP1, PAR, IW, LIW, W, LW)                HESD 067
      TEMP = TEMP - TEMP1                                                 HESD 068
      PAR(IND2) = PARI - STEP                                             HESD 069
```

```
      CALL FUNCT(0, NPAR, PAR, TEMP1, PAR, IW, LIW, W, LW)              HESD 070
      TEMP = TEMP + TEMP1                                               HESD 071
      PAR(IND3) = PARJ + STEP                                           HESD 072
      CALL FUNCT(0, NPAR, PAR, TEMP1, PAR, IW, LIW, W, LW)              HESD 073
      H(I,J) = (TEMP - TEMP1) / DENOM                                   HESD 074
      PAR(IND2) = PARI                                                  HESD 075
      PAR(IND3) = PARJ                                                  HESD 076
   40 CONTINUE                                                          HESD 077
C                                                                       HESD 078
C         FIND DIFFERENCE, SAVE OLD HESSIAN IN SAVE1.                   HESD 079
   50 TEMP = ZERO                                                       HESD 080
      DO 60 I = 1, NVAR                                                 HESD 081
      DO 60 J = 1, NVAR                                                 HESD 082
      PARI = H(I,J)                                                     HESD 083
      TEMP1 = ABS(PARI - SAVE1(I,J))                                    HESD 084
      PARJ = ABS(PARI)                                                  HESD 085
      IF (PARJ .GE. ONE) TEMP1 = TEMP1 / PARJ                           HESD 086
      TEMP = AMAX1(TEMP,TEMP1)                                          HESD 087
      SAVE1(I,J) = PARI                                                 HESD 088
   60 CONTINUE                                                          HESD 089
C                                                                       HESD 090
C         THIRD ITN, SAVE OLD HESSIAN IN SAVE2 (STEPLENGTH 1.0E-4).     HESD 091
      IF (ITER .NE. 3) GO TO 80                                         HESD 092
      DO 70 I = 1, NVAR                                                 HESD 093
      DO 70 J = 1, NVAR                                                 HESD 094
   70 SAVE2(I,J) = H(I,J)                                               HESD 095
C                                                                       HESD 096
C         TEST STOPPING CRITERION FOR ITERATION.                        HESD 097
   80 IF (TEMP .LT. TOL) GO TO 100                                      HESD 098
      IF (ITER .LT. 5) GO TO 20                                         HESD 099
C                                                                       HESD 100
C         NO CONVERGENCE IN 5 ITNS - USE SAVE2, WITH STEP TENM4.        HESD 101
      STEP = TENM4                                                      HESD 102
      DO 90 I = 1, NVAR                                                 HESD 103
      DO 90 J = 1, NVAR                                                 HESD 104
   90 H(I,J) = SAVE2(I,J)                                               HESD 105
C                                                                       HESD 106
C         EXIT, WRITE DETAILS (IND IS OUTPUT UNIT PASSED THROUGH IW),   HESD 107
C         AND SAVE THE NUMBER OF ITERATIONS REQUIRED.                   HESD 108
  100 IND = IW(4)                                                       HESD 109
      IW(1) = ITER                                                      HESD 110
      IF (IND .GT. 0) WRITE (IND,1000) ITER, STEP                       HESD 111
C                                                                       HESD 112
 1000 FORMAT (16H0HESSIAN DONE IN, I3, 6H ITNS,, 16H STEPSIZE USED =,   HESD 113
     *E11.3)                                                            HESD 114
C                                                                       HESD 115
      RETURN                                                            HESD 116
      END                                                               HESD 117
```

MASTER COPY - FOR REPRODUCTION PURPOSES

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD-A178 542

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ARO-15	2. GOVT ACCESSION NO. N/A	3. RECIPIENT'S CATALOG NUMBER N/A
4. TITLE (and Subtitle) A Family of Algorithms for the Estimation of the Parameters of the Stable Laws and the Parameters of Attracting Stable Laws		5. TYPE OF REPORT & PERIOD COVERED Working Paper 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) T.A. Delehanty A.S. Paulson		8. CONTRACT OR GRANT NUMBER(s) DAAG29-81-K-0110
9. PERFORMING ORGANIZATION NAME AND ADDRESS Rensselaer Polytechnic Institute Troy, New York 12180		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office Post Office Box 12211 Research Triangle Park, NC 27709		12. REPORT DATE 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

stable laws, parameter estimation, adaptive estimation, empirical characteristic function, domains of attraction, sensitivity analysis, nonlinear optimization, constrained estimation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper presents several families of algorithms for estimation of the parameters of the stable laws and the parameters of attracting stable laws. The paper also presents algorithms for estimation of the parameters of stable regression and stable autoregression models.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

DAT
FILM
5-